AWS Neuron

AWS

Dec 19, 2022

OVERVIEW

1 Neuron Quick Links 3

2 Get Started with PyTorch Neuron 5

3 Get Started with TensorFlow Neuron 11

4 Get Started with Apache MXNet (Incubating) Neuron 15

5 Neuron Performance 19

6 What's New 21

7 Annoucements 25

8 PyTorch Neuron 27

9 TensorFlow Neuron 957

10 MXNet Neuron 1743

11 Neuron Runtime 1997

12 Neuron Compiler 2019

13 Neuron Tools 2055

14 Setup Guide 2143

15 Deploy Containers with Neuron 2145

16 Developer Flows 2179

17 Neuron Architecture 2207

18 Neuron Features 2227

19 Neuron Application Notes 2241

20 Neuron FAQ 2283

21 Troubleshooting Guide 2285

22 Release Details 2287

23 Roadmap 2289

24 Support 2291

Python Module Index 2295

Index 2297

AWS Neuron is the SDK used to run deep learning workloads on AWS Inferentia and AWS Trainium based instances. It supports customers in their end-to-end ML development lifecycle to build new models, train and optimize these models, and then deploy them for production. To learn about the model architectures currently supported on Inf1 and Trn1 instances, please see *Model Architecture Fit Guidelines*. To learn about upcoming capabilities, please view the *Roadmap*.

AWS Neuron includes a deep learning compiler, runtime, and tools that are natively integrated into TensorFlow, PyTorch and Apache MXNet (incubating). The EC2 Trn1 instances are optimized for the highest performance and best price-performance training in AWS. The EC2 Inf1 instances are designed for high-performance deep learning inference applications. With Neuron, customers can quickly start using Inf/Trn instances through services like Amazon Sagemaker, Amazon Elastic Container Service (ECS), Amazon Elastic Kubernetes Service (EKS), AWS Batch, and AWS Parallel Cluster.

Check *Annoucements* and check *Neuron 2.6.0 (12/12/2022)* for latest release.

Get Started with PyTorch Neuron Get Started with TensorFlow Neuron Neuron Quick Links

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

ONE

NEURON QUICK LINKS

Overview

- *Get Started with PyTorch Neuron*
- *Get Started with TensorFlow Neuron*
- *Get Started with Apache MXNet (Incubating) Neuron*
- *Neuron Performance*
- *What's New*
- *Annoucements*

Neuron Samples

- PyTorch Neuron training samples for Trainium (torch-neuronx)
- Megatron-LM sample for Trainium
- AWS ParallelCluster samples for Trainium
- PyTorch Neuron inference samples for Inferentia (torch-neuron)
- TensorFlow Neuron inference samples for Inferentia (tensorflow-neuron)

ML Frameworks

- *PyTorch Neuron*
- *TensorFlow Neuron*
- *MXNet Neuron*

User Guide

- *Neuron Runtime*
- *Neuron Compiler*
- *Neuron Tools*
- *Setup Guide*
- *Deploy Containers with Neuron*
- *Developer Flows*

Learn Neuron

- *Neuron Architecture*
- *Neuron Features*
- *Neuron Application Notes*
- *Neuron FAQ*
- *Troubleshooting Guide*

About Neuron

- *Release Details*
- *Roadmap*
- *Support*

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWO

GET STARTED WITH PYTORCH NEURON

This page provide links that will assist you to quickly start with *PyTorch Neuron* for both Inference and Training.

Get Started with Training

Launch Trn1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Trn1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Trn1 instances sizes and pricing see Trn1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

Install Drivers and Tools

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
→PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install git
sudo yum install git -y


# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.*  -y

export PATH=/opt/aws/neuron/bin:$PATH
```

Install PyTorch Neuron (`torch-neuronx`)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
pip install -U pip

# Install packages from repos
```

(continues on next page)

(continued from previous page)

```
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install wget, awscli
pip install wget
pip install awscli

# Install Neuron packages
pip install torch-neuronx==1.12.0.1.*
pip install neuronx-cc==2.*
```

Run Tutorial

Multi-Layer Perceptron Training Tutorial

Visit PyTorch Neuron section for more

Get Started with Inference

Launch Inf1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Install Drivers and Tools

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##############################################################################################
↪#######################
# Remove older versions of Neuron
##############################################################################################
↪#######################
sudo yum remove aws-neuron-dkms -y
```

(continues on next page)

(continued from previous page)

```
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuron-tools -y
sudo yum remove aws-neuronx-tools -y

##########################################################################################
# To install or update to Neuron versions 2.5 and newer from previous releases:
# - DO NOT skip 'aws-neuronx-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH
```

Install PyTorch Neuron (`torch-neuron`)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.

sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip

# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron PyTorch)"
pip install jupyter notebook
pip install environment_kernels

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron neuron-cc[tensorflow] "protobuf<4" torchvision
```

Run Tutorial

ResNet-50

Visit PyTorch Neuron section for more

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

THREE

GET STARTED WITH TENSORFLOW NEURON

This page provide links that will assist you to quickly start with *TensorFlow Neuron*.

Get Started with Inference

Launch Inf1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Install Drivers and Tools

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# Remove older versions of Neuron
##########################################################################################
↪#######################
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuron-tools -y
```

(continues on next page)

(continued from previous page)

```
sudo yum remove aws-neuronx-tools -y

##########################################################################################
# To install or update to Neuron versions 2.5 and newer from previous releases:
# - DO NOT skip 'aws-neuronx-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms-2.* -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools=2.* -y

export PATH=/opt/aws/neuron/bin:$PATH
```

Install TensorFlow Neuron (`tensorflow-neuron`)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Install Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf==3.20.1"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuron -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron
```

Run Tutorial

ResNet-50

Visit TensorFlow Neuron section for more

Get Started with Training

Note: TensorFlow Neuron support for training workloads is coming soon.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`

CHAPTER

FOUR

GET STARTED WITH APACHE MXNET (INCUBATING) NEURON

This page provide links that will assist you to quickly start with *MXNet Neuron* (supporting inference only).

Get Started with Inference

Launch Inf1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Install Drivers and Tools

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# Remove older versions of Neuron
##########################################################################################
↪#######################
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuron-tools -y
```

(continues on next page)

(continued from previous page)

```
sudo yum remove aws-neuronx-tools -y

##########################################################################################
# To install or update to Neuron versions 2.5 and newer from previous releases:
# - DO NOT skip 'aws-neuronx-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms-2.* -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuronx-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH
```

Install MXNet Neuron

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Install Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Run Tutorial

ResNet-50

Visit MXNet Neuron section for more

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

FIVE

NEURON PERFORMANCE

The Neuron Performance pages provide a reference to the expected performance of various open-source models for popular deep learning in Natural Language Processing (NLP), Computer Vision (CV) and Recommender model tasks. We have included with each model links to allow you to setup and reconstruct the test with a few steps.

Inf1 Performance Numbers

Trn1 Performance Numbers

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

SIX

WHAT'S NEW

Table of contents

- *Neuron 2.6.0 (12/12/2022)*
- *Previous Releases*

6.1 Neuron 2.6.0 (12/12/2022)

Table of contents

- *What's New*
- *Neuron Components Release Notes*
  - *Inf1 and Trn1 common packages*
  - *Trn1 only packages*
  - *Inf1 only packages*

6.1.1 What's New

This release introduces the support of PyTorch 1.12 version, and introduces PyTorch Neuron (`torch-neuronx`) profiling through Neuron Plugin for TensorBoard. Pytorch Neuron (`torch-neuronx`) users can now profile their models through the following TensorBoard views:

- Operator Framework View
- Operator HLO View
- Operator Trace View

This release introduces the support of LAMB optimizer for FP32 mode, and adds support for *capturing snapshots* of inputs, outputs and graph HLO for debugging.

In addition, this release introduces the support of new operators and resolves issues that improve stability for Trn1 customers.

For more detailed release notes of the new features and resolved issues, see *Neuron Components Release Notes*.

6.1.2 Neuron Components Release Notes

Inf1 and Trn1 common packages

Component	Instance/s	Package/s	Details
Neuron Runtime	Trn1 , Inf1	• Trn1: `aws-neuronx-runtime-lib` (.deb, .rpm) • Inf1: Runtime is linked into the ML frameworks packages	• *Neuron Runtime Release Notes*
Neuron Runtime Driver	Trn1, Inf1	• `aws-neuronx-dkms` (.deb, .rpm)	• *Neuron Driver Release Notes*
Neuron System Tools	Trn1, Inf1	• `aws-neuronx-tools` (.deb, .rpm)	• *Neuron System Tools*
Containers	Trn1, Inf1	• `aws-neuronx-k8-plugin` (.deb, .rpm) • `aws-neuronx-k8-scheduler` (.deb, .rpm) • `aws-neuronx-oci-hooks` (.deb, .rpm)	• *Neuron K8 Release Notes* • *Neuron Containers Release Notes*
NeuronPerf (Inference only)	Trn1, Inf1	• `neuronperf` (.whl)	• neuronperf_rn

Trn1 only packages

Component	Instance/s	Package/s	Details
PyTorch Neuron	Trn1	• `torch-neuronx` (.whl)	• *PyTorch Neuron (torch-neuronx) release notes* • *PyTorch Neuron (torch-neuronx) - Training Supported Operators*
Neuron Compiler (Trn1 only)	Trn1	• `neuronx-cc` (.whl)	• *Neuron Compiler (neuronx-cc) release notes*
Collective Communication library	Trn1	• `aws-neuronx-collective` (.deb, .rpm)	• *Neuron Collectives Release Notes*

Note: In next releases `aws-neuronx-tools` and `aws-neuronx-runtime-lib` will add support for Inf1.

Inf1 only packages

Component	Instance/s	Package/s	Details
PyTorch Neuron	Inf1	• `torch-neuron` (.whl)	• *PyTorch Neuron (torch-neuron) release notes* • *PyTorch Neuron (torch-neuron) Supported operators*
TensorFlow Neuron	Inf1	• `tensorflow-neuron` (.whl)	• *TensorFlow Neuron (tensorflow-neuron (TF1.x)) Release Notes* • *TensorFlow Neuron (tensorflow-neuron (TF1.x)) Supported operators*
TensorFlow Model Server Neuron	Inf1	• `tensorflow-model-server-neuronx` (.deb, .rpm)	• tensorflow-modelserver-rn
Apache MXNet (Incubating)	Inf1	• `mx_neuron` (.whl)	• *Apache MXNet Neuron (Incubating) Release Notes* • *Neuron Apache MXNet (Incubating) Supported operators*
Neuron Compiler (Inf1 only)	Inf1	• `neuron-cc` (.whl)	• *Neuron Compiler (neuron-cc) for Inf1 Release Notes* • *Neuron Supported operators*

6.2 Previous Releases

- prev-rn
- prev-n1-rn

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

SEVEN

ANNOUCEMENTS

This page will be replaced by ABlog. It's here to make sure it's in the TOC.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

EIGHT

PYTORCH NEURON

PyTorch Neuron unlocks high-performance and cost-effective deep learning acceleration on AWS Trainium-based and Inferentia-based Amazon EC2 instances.

PyTorch Neuron plugin architecture enables native PyTorch models to be accelerated on Neuron devices, so you can use your existing framework application and get started easily with minimal code changes.

Training

Setup Guide for Trn1

This document is relevant for: `Trn1`

8.1 Install PyTorch Neuron (`torch-neuronx`)

Table of Contents

- *Develop on Trn1 instance*

8.1.1 Develop on Trn1 instance

PyTorch 1.12.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates
. /etc/os-release

sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
```

(continues on next page)

(continued from previous page)

```
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install git
sudo apt-get install git -y


# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo apt-get remove aws-neuron-dkms  -y
sudo apt-get remove aws-neuronx-dkms  -y
sudo apt-get remove aws-neuronx-oci-hook  -y
sudo apt-get remove aws-neuronx-runtime-lib -y
sudo apt-get remove aws-neuronx-collectives -y
sudo apt-get install aws-neuronx-dkms=2.* -y
sudo apt-get install aws-neuronx-oci-hook=2.* -y
sudo apt-get install aws-neuronx-runtime-lib=2.* -y
sudo apt-get install aws-neuronx-collectives=2.* -y

# Install EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo apt-get remove aws-neuron-tools  -y
sudo apt-get remove aws-neuronx-tools  -y
sudo apt-get install aws-neuronx-tools=2.* -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt install python3.8-venv
python3.8 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip

# Install wget, awscli
pip install wget
pip install awscli

# Install packages from  repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.12.0.1.*" "neuronx-cc==2.*" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y

# Install git
sudo yum install git -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.*  -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
python -m pip install -U pip

# Install wget, awscli
pip install wget
pip install awscli

# Install packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.12.0.1.*" "neuronx-cc==2.*" torchvision
```

PyTorch 1.11.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates
. /etc/os-release

sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install git
sudo apt-get install git -y


# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo apt-get remove aws-neuron-dkms  -y
sudo apt-get remove aws-neuronx-dkms  -y
sudo apt-get remove aws-neuronx-oci-hook  -y
sudo apt-get remove aws-neuronx-runtime-lib -y
sudo apt-get remove aws-neuronx-collectives -y
sudo apt-get install aws-neuronx-dkms=2.* -y
sudo apt-get install aws-neuronx-oci-hook=2.* -y
sudo apt-get install aws-neuronx-runtime-lib=2.* -y
sudo apt-get install aws-neuronx-collectives=2.* -y

# Install EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo apt-get remove aws-neuron-tools  -y
sudo apt-get remove aws-neuronx-tools  -y
sudo apt-get install aws-neuronx-tools=2.* -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt install python3.8-venv
python3.8 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
pip install -U pip

# Install wget, awscli
pip install wget
pip install awscli

# Install packages from  repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.*" "neuronx-cc==2.*" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y

# Install git
sudo yum install git -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
```

(continues on next page)

(continued from previous page)

```
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.*  -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
python -m pip install -U pip

# Install wget, awscli
pip install wget
pip install awscli

# Install packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.*" "neuronx-cc==2.*" torchvision
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.2 Update to latest PyTorch Neuron (torch-neuronx)

PyTorch 1.12.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Update OS packages
sudo apt-get update -y


# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver and Runtime
sudo apt-get install aws-neuronx-dkms=2.* -y
sudo apt-get install aws-neuronx-oci-hook=2.* -y
sudo apt-get install aws-neuronx-runtime-lib=2.* -y
sudo apt-get install aws-neuronx-collectives=2.* -y

# Update EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools=2.* -y

# Activate a Python virtual environment where Neuron pip packages were installed.
source aws_neuron_venv_pytorch_p38/bin/activate

# Install packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Update Python packages - Transformers package is needed for BERT
python -m pip install -U torch-neuronx=="1.12.0.1.*" neuronx-cc=="2.*" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y


# Update Neuron Driver and Runtime
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Update EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Update Neuron Tools
sudo yum install aws-neuronx-tools-2.*  -y

# Activate a Python virtual environment where Neuron pip packages were installed.
source aws_neuron_venv_pytorch_p37/bin/activate

# Update packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Update Python packages - Transformers package is needed for BERT
python -m pip install -U torch-neuronx=="1.12.0.1.*" neuronx-cc=="2.*" torchvision
```

PyTorch 1.11.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Update OS packages
sudo apt-get update -y


# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver and Runtime
sudo apt-get install aws-neuronx-dkms=2.* -y
sudo apt-get install aws-neuronx-oci-hook=2.* -y
sudo apt-get install aws-neuronx-runtime-lib=2.* -y
sudo apt-get install aws-neuronx-collectives=2.* -y

# Update EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools=2.* -y

# Activate a Python virtual environment where Neuron pip packages were installed.
source aws_neuron_venv_pytorch_p38/bin/activate

# Install packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Update Python packages - Transformers package is needed for BERT
python -m pip install -U torch-neuronx=="1.11.0.1.*" neuronx-cc=="2.*" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y


# Update Neuron Driver and Runtime
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Update EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Update Neuron Tools
sudo yum install aws-neuronx-tools-2.*  -y

# Activate a Python virtual environment where Neuron pip packages were installed.
source aws_neuron_venv_pytorch_p37/bin/activate

# Update packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Update Python packages - Transformers package is needed for BERT
python -m pip install -U torch-neuronx=="1.11.0.1.*" neuronx-cc=="2.*" torchvision
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.3 Install Previous PyTorch Neuron Releases (torch-neuronx)

This document is relevant for: Trn1

8.3.1 Install PyTorch Neuron (Neuron 2.4.0)

PyTorch 1.11.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates
. /etc/os-release

sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y


# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo apt-get remove aws-neuron-dkms  -y
sudo apt-get remove aws-neuronx-dkms  -y
sudo apt-get remove aws-neuronx-oci-hook  -y
sudo apt-get remove aws-neuronx-runtime-lib -y
sudo apt-get remove aws-neuronx-collectives -y
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y
sudo apt-get install aws-neuronx-oci-hook=2.1.2.0 -y
sudo apt-get install aws-neuronx-runtime-lib=2.10.15.0 -y
sudo apt-get install aws-neuronx-collectives=2.10.17.0 -y

# Install EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
```

(continues on next page)

(continued from previous page)

```
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo apt-get remove aws-neuron-tools  -y
sudo apt-get remove aws-neuronx-tools  -y
sudo apt-get install aws-neuronx-tools=2.5.16.0 -y

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt install python3.8-venv
python3.8 -m venv aws_neuron_venv_pytorch_p38
source aws_neuron_venv_pytorch_p38/bin/activate
pip install -U pip

# Install packages from beta repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.2.0" "neuronx-cc==2.2.0.73" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.6.5.0  -y
sudo yum install aws-neuronx-oci-hook-2.1.2.0  -y
sudo yum install aws-neuronx-runtime-lib-2.10.15.0  -y
sudo yum install aws-neuronx-collectives-2.10.17.0  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.5.16.0  -y

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch_p37
source aws_neuron_venv_pytorch_p37/bin/activate
python -m pip install -U pip

# Install packages from beta repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.2.0" "neuronx-cc==2.2.0.73" torchvision
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.3.2 Install PyTorch Neuron (Neuron 2.3.0)

PyTorch 1.11.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates
. /etc/os-release

sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y


# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo apt-get remove aws-neuron-dkms  -y
sudo apt-get remove aws-neuronx-dkms  -y
sudo apt-get remove aws-neuronx-oci-hook  -y
sudo apt-get remove aws-neuronx-runtime-lib -y
sudo apt-get remove aws-neuronx-collectives -y
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y
sudo apt-get install aws-neuronx-oci-hook=2.0.16.0 -y
sudo apt-get install aws-neuronx-runtime-lib=2.9.64.0 -y
sudo apt-get install aws-neuronx-collectives=2.9.86.0 -y

# Install EFA Driver(only required for multiinstance training)

curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
```

(continues on next page)

(continued from previous page)

```
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo apt-get remove aws-neuron-tools  -y
sudo apt-get remove aws-neuronx-tools  -y
sudo apt-get install aws-neuronx-tools=2.4.14.0 -y

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt install python3.8-venv
python3.8 -m venv aws_neuron_venv_pytorch_p38
source aws_neuron_venv_pytorch_p38/bin/activate
pip install -U pip

# Install packages from beta repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.1.1" "neuronx-cc==2.1.0.76" torchvision
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y


# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
```

(continues on next page)

(continued from previous page)

```
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.5.41.0  -y
sudo yum install aws-neuronx-oci-hook-2.0.16.0  -y
sudo yum install aws-neuronx-runtime-lib-2.9.64.0  -y
sudo yum install aws-neuronx-collectives-2.9.86.0  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.4.14.0  -y

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch_p37
source aws_neuron_venv_pytorch_p37/bin/activate
python -m pip install -U pip

# Install packages from beta repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.1.1" "neuronx-cc==2.1.0.76" torchvision
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

Tutorials (`torch-neuronx`)

This document is relevant for: `Trn1`

8.4 Hugging Face BERT Pretraining Tutorial

This tutorial explains how to run Hugging Face BERT-Large model pretraining on Trainium using PyTorch Neuron.

The Hugging Face BERT pretraining example demonstrates the steps required to perform single-node, multi-accelerator PyTorch model training using the new AWS EC2 Trn1 (Trainium) instances and the AWS Neuron SDK. This tutorial is an adaptation of an existing BERT example with the following important characteristics:

- Framework: PyTorch/XLA
- Model: Hugging Face BertForPreTraining
- Optimizer: AdamW
- Scheduler: Hugging Face's get_linear_schedule_with_warmup
- Allreduce occurs before optimizer step, after gradient accumulations (following DeepSpeed's Smart Gradient Accumulation)

As done in the original BERT paper, BERT pretraining happens in two phases. In the first phase (phase 1) BERT maximum sequence length is fixed at 128 tokens, while in phase 2 it is fixed at 512 tokens.

Neuron provides access to Trainium devices through an extension of PyTorch/XLA - a library that includes the familiar PyTorch interface along with XLA-specific additions. For additional details relating to PyTorch/XLA, please refer to the official PyTorch/XLA documentation.

Table of Contents

- *Phase 1 BERT-Large pretraining*
  - *Setting up the training environment on trn1.32xlarge*
  - *Downloading tokenized and sharded dataset files*
  - *Number of workers*
  - *BFloat16 and stochastic rounding in phase 1*
  - *Pre-compilation*
  - *Initiating a Training Job*
  - *Monitoring Progress of the Training Job*
  - *Monitoring Training Job Progress using neuron-top*
  - *Monitoring Training Job Progress using TensorBoard*
  - *Finishing the tutorial*
- *Phase 1 BERT-Large pretraining on two instances*
- *Phase 2 BERT-Large pretraining*
  - *Training Environment*
  - *Initiating a Training Job*

- *Tools*
  - *neuron-ls*
  - *neuron-top*
  - *Generating tokenized and sharded dataset files*
- *Known issues and limitations*
  - *NaNs seen with transformers version >= 4.21.0 when running HF BERT fine-tuning or pretraining with XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1*
  - *BERT-large compilation limitations*
  - *BERT-large pretraining with pretokenized dataset hangs when using xm.save*
  - *BERT-large two worker pretraining hangs or run out of host memory during checkpointing on trn1.2xlarge*
  - *BERT precompilation using neuron_parallel_compile hangs when using torchrun*
  - *Reduced multi-node performance with Neuron PyTorch 1.12 (release 2.6)*
- *Troubleshooting*
  - *ModuleNotFoundError: No module named 'torch' , 'torch_xla', 'transformers', etc*

Note: Logs used in tutorials do not present latest performance numbers

For latest performance numbers visit *Neuron Performance*

8.4.1 Phase 1 BERT-Large pretraining

Setting up the training environment on trn1.32xlarge

The BERT training script `dp_bert_large_hf_pretrain_hdf5.py` can run on a Trainium instance (trn1.32xlarge) that contains the appropriate Neuron runtime and Python dependencies.

First, on a trn1.32xlarge instance, follow the installation instructions at:

- *Install PyTorch Neuron on Trn1*

Please set the storage of instance to *512GB* or more if you intent to run multiple experiments and save many checkpoints.

For all the commands below, make sure you are in the virtual environment that you have created above before you run the commands:

```
source ~/aws_neuron_venv_pytorch/bin/activate
```

Next, download the Python-based training script `dp_bert_large_hf_pretrain_hdf5.py` and `requirements.txt` and install the requirements:

```
mkdir -p ~/examples/dp_bert_hf_pretrain
cd ~/examples/dp_bert_hf_pretrain
wget https://raw.githubusercontent.com/aws-neuron/aws-neuron-samples/master/torch-
↪neuronx/training/dp_bert_hf_pretrain/dp_bert_large_hf_pretrain_hdf5.py
wget https://raw.githubusercontent.com/aws-neuron/aws-neuron-samples/master/torch-
↪neuronx/training/dp_bert_hf_pretrain/requirements.txt
```

(continues on next page)

(continued from previous page)

```
python3 -m pip install -r requirements.txt
```

Downloading tokenized and sharded dataset files

To download the tokenized and sharded dataset files needed for this tutorial, please run the following commands:

```
mkdir -p ~/examples_datasets/
pushd ~/examples_datasets/
aws s3 cp s3://neuron-s3/training_datasets/bert_pretrain_wikicorpus_tokenized_hdf5/bert_
↪pretrain_wikicorpus_tokenized_hdf5_seqlen128.tar .  --no-sign-request
tar -xf bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128.tar
rm bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128.tar
aws s3 cp s3://neuron-s3/training_datasets/bert_pretrain_wikicorpus_tokenized_hdf5/bert_
↪pretrain_wikicorpus_tokenized_hdf5_seqlen512.tar .  --no-sign-request
tar -xf bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512.tar
rm bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512.tar
popd
```

`~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128` will now have the tokenized and sharded dataset files for phase 1 pretraining and `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512` for phase 2 pretraining.

Number of workers

You will be using torchrun (PyTorch's Elastic Launch) to run some of the commands in this tutorial. When running the training script, you can configure the number of NeuronCores to use for training by using torchrun's `--nproc_per_node` option. In this tutorial, we use 32 NeuronCores on trn1.32xlarge.

Note: Currently Neuron Runtime only support 1 and 2 worker configurations on trn1.2xlarge and 1, 2, 8, and 32-worker configurations on trn1.32xlarge.

BFloat16 and stochastic rounding in phase 1

Phase 1 pretraining performance can be increased by using full BFloat16 casting and stochastic rounding. Full BFloat16 casting and stochastic rounding can be enabled by setting environment variable `XLA_USE_BF16=1` when launching the pretraining job. `XLA_DOWNCAST_BF16=1` can also be used instead of `XLA_USE_BF16=1` to preserve part of the training loop in FP32. Here we use `XLA_DOWNCAST_BF16=1` to ensure smooth loss curve when loss averaging is used. To achieve maximum performance while maintaining loss convergence characteristics, we are using batch size of 16 and gradient accumulation microsteps of 32 to maintain global batch size of 16384 for phase 1. The batch size and gradient accumulation microstep changes can be set by launching the BERT pretraining script `dp_bert_large_hf_pretrain_hdf5.py` with command-line arguments `--batch_size=16 --grad_accum_usteps=32`, as seen in the following steps.

Pre-compilation

PyTorch Neuron evaluates operations lazily during execution of the training loops, which means it builds a symbolic graph in the background and the graph is executed in hardware only when the tensor is printed, transfered to CPU, or `xm.mark_step()` is encountered (`xm.mark_step()` is implicitly called by `pl.MpDeviceLoader/pl.ParallelLoader`). During execution of the training loops, PyTorch Neuron can build multiple graphs depending on the number of conditional paths taken. For BERT-Large pretraining, PyTorch Neuron builds multiple unique graphs that should be compiled before running on the NeuronCores. PyTorch Neuron will compile those graphs only if they are not in the XLA in-memory cache or the persistent cache. To reduce the compilation time of these graphs, you can pre-compile those graphs using the utility `neuron_parallel_compile` (provided by the `libneuronxla` package, a transitive dependency of `torch-neuronx`) as shown:

```
cd ~/examples/dp_bert_hf_pretrain
neuron_parallel_compile XLA_DOWNCAST_BF16=1 torchrun --nproc_per_node=32 \
dp_bert_large_hf_pretrain_hdf5.py \
--steps_this_run 10 \
--batch_size 16 \
--grad_accum_usteps 32 |& tee compile_log.txt
```

This command performs a fast trial run of the training script to build graphs and then do parallel compilations on those graphs using multiple processes of Neuron Compiler before populating the on-disk persistent cache with compiled graphs. This helps make the actual training run faster because the compiled graphs will loaded from the persistent cache. Currently it takes ~13 minutes to compile the BERT-Large model training step using the pre-compilation script (compare to ~40 minute if not using the pre-compilation script). Note that the command above specifies 32 NeuronCores for trn1.32xlarge via –nproc_per_node option.

The pretokenized dataset is expected to be at `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128/` by default (see above for downloading instructions) and can be changed via the `--data_dir` option.

Note: The trial run during pre-compilation currently outputs invalid loss numbers. Please disregard them.

Note: The command after `neuron_parallel_compile` should match the actual run command, except for the option `--steps_this_run` which shortens the trial run just enough to allow the tool to build all the graphs needed for the actual run.

If you interrupt the run and restart the execution without changing model configurations or training hyperparameters, the new run will detect the cached graphs in the persistent cache (on-disk) and reload the compiled graphs for execution, avoiding any recompilation time.

Changes made to the BERT model configuration (layers, hidden size, attention heads in the get_model function), batch size (using `--batch_size` option), optimizer or number of workers may trigger graph recompilation. It is best to rerun the pre-compilation step above if these changes are made.

You can adjust the following hyperparameters without changing the model and causing recompilation:

- Number of global steps to run (`--steps_this_run` option)
- Learning rate (`--lr` option)
- Gradient accumulation steps > 1 (`--grad_accum_usteps` option). If 1 then there's no gradient accumulation and the graphs change causing recompilation.

Initiating a Training Job

After running the pre-compilation step, continue with the actual phase 1 pretraining by running the following set of commands to launch 32 data parallel distributed training workers on trn1.32xlarge:

```
cd ~/examples/dp_bert_hf_pretrain
XLA_DOWNCAST_BF16=1 torchrun --nproc_per_node=32 \
dp_bert_large_hf_pretrain_hdf5.py \
--batch_size 16 \
--grad_accum_usteps 32 |& tee run_pretrain_log.txt
```

As the training script launches, you will initially see several console messages indicating that the Neuron Runtime is initializing:

```
Using Neuron Runtime
Using Neuron Runtime
Using Neuron Runtime
Using Neuron Runtime
Using Neuron Runtime
...
```

A few moments later, you will see the Training Configuration and Model Configuration in the output:

```
--------TRAINING CONFIG----------
Namespace(batch_size=16, data_dir='~/examples_datasets/
bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128/', debug=False,
enable_pt_autocast=False, grad_accum_usteps=32, local_rank=0, lr=0.0004,
max_pred_len=20, max_steps=28125, metrics_file='/tmp/test_dict.json',
minimal_ckpt=False, num_ckpts_to_keep=1, output_dir='./output',
phase1_end_step=28125, phase2=False, resume_ckpt=False, resume_step=-1,
seed=12349, seq_len=128, shards_per_ckpt=1, steps_this_run=28125, warmup_steps=2000)
```

```
--------MODEL CONFIG----------
BertConfig {
"_name_or_path": "bert-large-uncased",
"architectures": [
"BertForMaskedLM"
],
"attention_probs_dropout_prob": 0.1,
"classifier_dropout": null,
"gradient_checkpointing": false,
"hidden_act": "gelu",
"hidden_dropout_prob": 0.1,
"hidden_size": 1024,
"initializer_range": 0.02,
"intermediate_size": 4096,
"layer_norm_eps": 1e-12,
"max_position_embeddings": 512,
"model_type": "bert",
"num_attention_heads": 16,
"num_hidden_layers": 24,
"pad_token_id": 0,
"position_embedding_type": "absolute",
```

(continues on next page)

(continued from previous page)

```
"transformers_version": "4.15.0",
"type_vocab_size": 2,
"use_cache": true,
"vocab_size": 30522
}
```

As the worker processes begin training on the BERT dataset, you will begin to see training metrics and the learning rate logged to the console approximately every training step. The metrics include average_loss, step_loss, and throughput:

```
LOG Thu Sep 29 22:30:10 2022 - (0, 78) step_loss : 9.1875  learning_rate : 1.56e-05 ␣
↪throughput : 2873.14
LOG Thu Sep 29 22:30:16 2022 - (0, 79) step_loss : 8.9375  learning_rate : 1.58e-05 ␣
↪throughput : 2878.09
LOG Thu Sep 29 22:30:22 2022 - (0, 80) step_loss : 9.0000  learning_rate : 1.60e-05 ␣
↪throughput : 2875.31
LOG Thu Sep 29 22:30:27 2022 - (0, 81) step_loss : 9.0000  learning_rate : 1.62e-05 ␣
↪throughput : 2877.35
LOG Thu Sep 29 22:30:33 2022 - (0, 82) step_loss : 8.8750  learning_rate : 1.64e-05 ␣
↪throughput : 2872.55
LOG Thu Sep 29 22:30:39 2022 - (0, 83) step_loss : 9.0000  learning_rate : 1.66e-05 ␣
↪throughput : 2876.17
LOG Thu Sep 29 22:30:44 2022 - (0, 84) step_loss : 9.1250  learning_rate : 1.68e-05 ␣
↪throughput : 2872.48
LOG Thu Sep 29 22:30:50 2022 - (0, 85) step_loss : 9.0000  learning_rate : 1.70e-05 ␣
↪throughput : 2873.39
```

By default, the training script will store all output files under `~/examples/dp_bert_hf_pretrain/output`. The output files consist of the following:

- PyTorch model checkpoint files, with names containing the global step of the checkpoint (ckpt_2000.pt, ckpt_4000.pt, etc.). Currently, the training script saves a checkpoint after every dataset shard. The frequency of saving checkpoint can be reduced by increasing the number of dataset shards per checkpoint, using option `--shards_per_ckpt`. Furthermore, the number of checkpoints kept at a given time is limited by `--num_ckpts_to_keep` option (currently default to 1).
- TensorBoard log files (each training run will store its logs in a subdirectory with prefix `neuron_tblogs_`).

Monitoring Progress of the Training Job

Using a single Trn1 instance with 32 NeuronCores, the current BERT phase 1 pretraining will finish in about 45 hours. During this time, you will see the average loss metric begin at about 11.2 and ultimately converge to about 1.4.

Monitoring Training Job Progress using neuron-top

With the training job still running, launch a second SSH connection into the trn1 instance, and use the `neuron-top` command to examine the aggregate NeuronCore utilization. If you have not modified the `--nproc_per_node` option in the run command, you should observe that all 32 NeuronCores are participating in the training job, with utilization fluctuating around 80%.

Monitoring Training Job Progress using TensorBoard

The demo includes TensorBoard-compatible logging, which allows the learning rate and training metrics to be monitored in real-time. By default, the training script logs metrics to the following TensorBoard log directory `~/examples/dp_bert_hf_pretrain/output/neuron_tblogs_<date/time>_<training configs>`.

In order to view your training metrics in TensorBoard, first run the following commands in your SSH session:

```
cd ~/examples/dp_bert_hf_pretrain
tensorboard --logdir ./output
```

Once running, open a new SSH connection to the instance and port-forward TCP port 6006 (ex: `-L 6006:127.0.0.1:6006`). Once the tunnel is established, TensorBoard can then be accessed via web browser at the following URL: http://localhost:6006. Please note that you will not be able to access TensorBoard if you disconnect your port-forwarding SSH session to the Trainium instance.

Finishing the tutorial

Once you are ready, there are a couple of options for finishing the BERT pretraining demo:

1. **Allow the training script to run to completion**. If you would like to observe the training script run to completion, it is recommended to launch the training script from a terminal multiplexer such as `tmux` or `screen`, and then detach the session so that the training script can run in the background. With this approach, you can safely let the training script run unattended, without risk of an SSH disconnection causing the training job to stop running.
2. **Stop the training job early**. To stop the training job early, press CTRL-C in the terminal window in which you launched the training script. In some cases, if you manually cancel a job using CTRL-C and then later want to run the job again, you might first need to execute `sudo rmmod neuron; sudo modprobe neuron` in order to reload/reset the Neuron driver.

8.4.2 Phase 1 BERT-Large pretraining on two instances

If you have two trn1.32xlarge instances with EFA-enabled interfaces, using EFA-enabled security group, and setup using *Install PyTorch Neuron on Trn1*, you can run multi-instance BERT-Large pretraining. The following example demonstrate running BERT phase 1 pretraining on two instances. To ensure that the global batch size remains at 16384 for phase 1, the gradient accumulation microstep count is reduced by half when the number of instances is 2. NOTE: To run on multiple instances, you will need to use trn1.32xlarge instances and using all 32 NeuronCores on each instance.

On the rank-0 Trn1 host (root), run with `--node_rank=0` using torchrun utility, and `--master_addr` set to rank-0 host's IP address:

```
cd ~/examples/dp_bert_hf_pretrain
export FI_EFA_USE_DEVICE_RDMA=1
export FI_PROVIDER=efa
export BUCKET_CAP_MB=512
export XLA_TRANSFER_SEED_ASYNC=1
XLA_DOWNCAST_BF16=1 torchrun --nproc_per_node=32 --nnodes=2 --node_rank=0 --master_addr=
↪<root IP> --master_port=2020 \
dp_bert_large_hf_pretrain_hdf5.py \
--batch_size 16 \
--grad_accum_usteps 16 |& tee run_pretrain_log.txt
```

On another Trn1 host, run with `--node_rank=1`, and `--master_addr` also set to rank-0 host's IP address:

```
cd ~/examples/dp_bert_hf_pretrain
export FI_EFA_USE_DEVICE_RDMA=1
export FI_PROVIDER=efa
export BUCKET_CAP_MB=512
export XLA_TRANSFER_SEED_ASYNC=1
XLA_DOWNCAST_BF16=1 torchrun --nproc_per_node=32 --nnodes=2 --node_rank=1 --master_addr=
↪<root IP> --master_port=2020 \
dp_bert_large_hf_pretrain_hdf5.py \
--batch_size 16 \
--grad_accum_usteps 16 |& tee run_pretrain_log.txt
```

It is important to launch rank-0 worker with `--node_rank=0` to avoid hang.

To train on multiple instances, it is recommended to use a ParallelCluster. For a ParallelCluster example, please see Train a model on AWS Trn1 ParallelCluster.

8.4.3 Phase 2 BERT-Large pretraining

As mentioned above, BERT pretraining happens in two phases. In phase 1, the sequence length is 128. In phase 2, the sequence length increases to 512. This additional training phase will further reduce the pretraining loss and improve the metrics for the fine-tune tasks that usually follow. The setup is very similar to the phase 1, with some differences in training environment and command line arguments highlighted below.

Training Environment

The following dataset and checkpoint are required:

- `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512` is WikiCorpus training dataset that is preprocessed (tokenized and pre-masked) for phase 2.
- `~/examples/dp_bert_hf_pretrain/output/ckpt_<phase1_end_step>.pt` is the final checkpoint from phase 1. It's generated automatically at the end of phase 1 pretraining. For convenience, one can also download the example available at `s3://neuron-s3/training_checkpoints/pytorch/dp_bert_large_hf_pretrain/ckpt_28125.pt`, which is collected after 28125 training steps in phase 1. Phase 2 will continue training by loading this checkpoint. During its progression, phase 2 continues to generate its own checkpoints in output directory, following the naming convention `ckpt_<global_steps>.pt`

Initiating a Training Job

To launch the phase 2 pretraining job, run the same python script `dp_bert_large_hf_pretrain_hdf5.py` as before except with different options for phase 2. Again, we are using BFloat16 casting and stochastic rounding by setting environment variable `XLA_DOWNCAST_BF16=1`. For phase 2, we are using global batch size of 32768, with worker device batch size of 2 and gradient accumulation microsteps of 512. The pretokenized dataset is expected to be at `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512/` following the setup steps above and is set via `--data_dir` option.

```
cd ~/examples/dp_bert_hf_pretrain
XLA_DOWNCAST_BF16=1 torchrun --nproc_per_node=32 dp_bert_large_hf_pretrain_hdf5.py \
    --data_dir ~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512/ \
    --lr 2.8e-4 \
    --phase2 \
    --resume_ckpt \
    --phase1_end_step 28125 \
    --batch_size 2 \
    --grad_accum_usteps 512 \
    --seq_len 512 \
    --max_pred_len 80 \
    --warmup_steps 781 \
    --max_steps 1563 \
    |& tee run_pretrain_log_phase2.txt
```

The output below is expected as the job is initiated. Step 28125 is the phase1_end_step in this run, which could be different if phase1 training stops at a different global step.

```
Worker 21 resuming from checkpoint ./output/ckpt_28125.pt at step 28125
Worker 23 resuming from checkpoint ./output/ckpt_28125.pt at step 28125
Worker 27 resuming from checkpoint ./output/ckpt_28125.pt at step 28125
Worker 26 resuming from checkpoint ./output/ckpt_28125.pt at step 28125
Worker 20 resuming from checkpoint ./output/ckpt_28125.pt at step 28125
```

(continues on next page)

(continued from previous page)

```
Worker 22 resuming from checkpoint ./output/ckpt_28125.pt at step 28125

--------TRAINING CONFIG----------
Namespace(batch_size=2, data_dir='/home/ec2-user/examples_datasets/
bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512/', debug=False,
enable_pt_autocast=False, grad_accum_usteps=512, local_rank=0, lr=0.0002,
max_pred_len=80, max_steps=28125, metrics_file='/tmp/test_dict.json',
minimal_ckpt=False, num_ckpts_to_keep=1, output_dir='./output',
phase1_end_step=28125, phase2=True, resume_ckpt=True, resume_step=-1,
seed=12349, seq_len=512, shards_per_ckpt=1, steps_this_run=32, warmup_steps=781)

--------MODEL CONFIG----------
BertConfig {
  "_name_or_path": "bert-large-uncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.15.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

As the phase 2 training proceeds, similar metrics to phase 1 will appear on the console, showing the loss, learning rate, and throughput:

```
LOG Tue Sep 27 20:56:35 2022 - (0, 26) step_loss : 4.3438  learning_rate : 6.66e-06 ␣
↪throughput : 494.55
LOG Tue Sep 27 20:57:40 2022 - (0, 27) step_loss : 4.0938  learning_rate : 6.91e-06 ␣
↪throughput : 495.67
LOG Tue Sep 27 20:58:46 2022 - (0, 28) step_loss : 4.1875  learning_rate : 7.17e-06 ␣
↪throughput : 496.18
LOG Tue Sep 27 20:59:53 2022 - (0, 29) step_loss : 4.0000  learning_rate : 7.43e-06 ␣
↪throughput : 495.31
LOG Tue Sep 27 21:00:58 2022 - (0, 30) step_loss : 4.2500  learning_rate : 7.68e-06 ␣
↪throughput : 495.60
LOG Tue Sep 27 21:02:05 2022 - (0, 31) step_loss : 4.3125  learning_rate : 7.94e-06 ␣
↪throughput : 495.50
```

(continues on next page)

(continued from previous page)

```
LOG Tue Sep 27 21:03:10 2022 - (0, 32) step_loss : 4.4688  learning_rate : 8.19e-06 ␣
↪throughput : 496.02
```

8.4.4 Tools

While running the tutorial, try experimenting with the following Neuron tools, which help monitor and evaluate compute utilization in real-time:

neuron-ls

The `neuron-ls` command describes the number of Neuron devices present in the system, along with the associated NeuronCore count, memory, and PCI device information:

```
[ec2-user@ip-10-0-1-68 ~]$ neuron-ls
Neuron Tools have changed with Neuron v1.16.0. Go here to learn more:
https://awsdocs-neuron.readthedocs-hosted.com/en/latest/neuron-guide/neuron-tools/

If you're still running Neuron Runtime 1.0, run this program as sudo to display neuron-rtd
runtime information

+--------+--------+--------+---------+
| NEURON | NEURON | NEURON |   PCI   |
| DEVICE | CORES  | MEMORY |   BDF   |
+--------+--------+--------+---------+
| 0      | 2      | 32 GB  | 10:1c.0 |
| 1      | 2      | 32 GB  | 20:1b.0 |
| 2      | 2      | 32 GB  | 10:1d.0 |
| 3      | 2      | 32 GB  | 20:1c.0 |
| 4      | 2      | 32 GB  | a0:1d.0 |
| 5      | 2      | 32 GB  | 90:1c.0 |
| 6      | 2      | 32 GB  | a0:1c.0 |
| 7      | 2      | 32 GB  | 90:1b.0 |
| 8      | 2      | 32 GB  | 10:1e.0 |
| 9      | 2      | 32 GB  | 20:1d.0 |
| 10     | 2      | 32 GB  | 10:1b.0 |
| 11     | 2      | 32 GB  | 20:1e.0 |
| 12     | 2      | 32 GB  | a0:1b.0 |
| 13     | 2      | 32 GB  | 90:1e.0 |
| 14     | 2      | 32 GB  | a0:1e.0 |
| 15     | 2      | 32 GB  | 90:1d.0 |
+--------+--------+--------+---------+
[ec2-user@ip-10-0-1-68 ~]$
```

You will find that the Trn1 instance has 16 Neuron devices, each with 2 NeuronCores. This configuration allows you to train the model using a total of 32 workers, one per NeuronCore, within a single instance.

Additional information regarding neuron-ls can be found in the neuron-ls user guide.

neuron-top

The `neuron-top` command presents a high-level view of the Neuron environment, including the utilization of each of the NeuronCores, any models that are currently loaded onto one or more NeuronCores, process IDs for any processes that are leveraging the Neuron runtime, and basic system statistics relating to vCPU and memory usage.

Please note that `neuron-top` can either display aggregate NeuronCore utilization for 'all' processes (the default), or alternatively display the NeuronCore utilization for a particular process. You can toggle through the aggregate and per-process views using the `a` and `d` keys. The screenshot below illustrates the default aggregate view:

Please refer to the neuron-top user guide for additional details.

Generating tokenized and sharded dataset files

This section is for generating tokenized and sharded dataset files from WikiCorpus dataset. If you just want the pre-genenerated dataset files, please see `Downloading tokenized and sharded dataset files` section above.

On a c5n.18xlarge instance launched with Deep Learning Conda AMI and 512GB disk space, you can generate the preprocessed datasets from WikiCorpus dataset using NVidia's DeepLearningExamples for BERT pretraining. The preprocessing converts the WikiCorpus dataset to tokenized data and shard the data into multiple shards for parallel loading. The full flow takes about 8.7 hours:

```
source activate pytorch_latest_p37
cd ~/
git clone https://github.com/NVIDIA/DeepLearningExamples.git
cd DeepLearningExamples
git checkout 81b9010096b6f9812e3977b607669f6ec8b16561
sudo mkdir -m a=rwx /workspace
cp -rf PyTorch/LanguageModeling/BERT /workspace/bert
cd /workspace
git clone https://github.com/attardi/wikiextractor.git
cd wikiextractor
git checkout 6408a430fc504a38b04d37ce5e7fc740191dee16
cd /workspace/bert
# increase num processes and shards
ex -s "+%s/\(bertPrep\.py\)\( --action create_hdf5_files\)/\1 --n_processes 32 --n_test_
↪shards 1024 --n_training_shards 1024\2" "+wq" data/create_datasets_from_start.sh
export BERT_PREP_WORKING_DIR=/workspace/data/
time ./data/create_datasets_from_start.sh wiki_only |& tee log
```

After execution is finished, phase 1 pre-tokenized and sharded dataset is located at:

`/workspace/data/hdf5_lower_case_1_seq_len_128_max_pred_20_masked_lm_prob_0.15_random_seed_12345_dupe_factor_5/wikicorpus_en/`

Copy this entire directory to `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen128` of the trn1.32xlarge machine.

Phase 2 pre-tokenized dataset is located at:

`/workspace/data/hdf5_lower_case_1_seq_len_512_max_pred_80_masked_lm_prob_0.15_random_seed_12345_dupe_factor_5/wikicorpus_en/`

Copy this entire directory to `~/examples_datasets/bert_pretrain_wikicorpus_tokenized_hdf5_seqlen512` of the trn1.32xlarge machine.

8.4.5 Known issues and limitations

NaNs seen with transformers version >= 4.21.0 when running HF BERT fine-tuning or pretraining with XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1

When running HuggingFace BERT (any size) fine-tuning tutorial or pretraining tutorial with transformers version >= 4.21.0 and using XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1, you will see NaNs in the loss immediately at the first step. More details on the issue can be found at pytorch/xla#4152. The workaround is to use 4.20.0 or earlier (the tutorials currently recommend version 4.15.0) or add `transformers.modeling_utils.get_parameter_dtype = lambda x: torch.bfloat16` to the Python script.

BERT-large compilation limitations

Optimal BERT-large phase 1 (sequence length 128) batch size is currently 8 for FP32 and 16 for full BF16 with stochastic rounding. Optimal BERT-large phase 2 (sequence length 512) batch size is currently 1 for FP32 and 2 for full BF16 with stochastic rounding.

BERT-large pretraining with pretokenized dataset hangs when using xm.save

Currently, BERT-large pretraining with pretokenized dataset hangs when `xm.save` is used outside of the main training loop.

```
Loop through HDF5 sharded dataset files:
    Train on one HDF5 sharded dataset file
        Loop through batched samples:
            Training iteration
    Save checkpoint using xm.save
```

The reason is that xm.save has a synchronization point. However, the HDF5 shared data files do not have the same number of training samples so the workers cannot all reach xm.save in the same iteration.

The workaround is to use `xm._maybe_convert_to_cpu` to ensure tensors are moved to CPU followed by `torch.save` as done in the BERT-large pretraining tutorial:

```
cpu_data = xm._maybe_convert_to_cpu(data)
```

BERT-large two worker pretraining hangs or run out of host memory during checkpointing on trn1.2xlarge

On trn1.2xlarge, where there's limited host memory and CPU resources, the BERT-large two worker pretraining may hang or run out of host memory during checkpointing. This problem can be worked around by not saving optimizer and LR scheduler states in the checkpoint. This is enabled by `--minimal_ckpt` option of the pretraining script.

BERT precompilation using neuron_parallel_compile hangs when using torchrun

We use neuron_parallel_compile in front of the short run command to do precompilation. However, the following command hangs when running BERT parallel compilation with torchrun:

```
neuron_parallel_compile XLA_USE_BF16=1 torchrun --nproc_per_node=32 --nnodes=1 dp_bert_
↪large_hf_pretrain_hdf5.py --steps_this_run 5

...
Updating train metrics in provide results.json file
Current data: {'num_workers': 32, 'epoch': 0, 'steps': 5, 'microsteps': 320, 'loss': -
↪22172234.0, 'train_time_minutes': 0.7424166639645894, 'throughput_average': 1839.
↪0391805624324, 'throughput_peak': 1840.0107059878164, 'batch_size': 8, 'max_length':␣
↪128}
Updating with data: {'num_workers': 32, 'epoch': 0, 'steps': 5, 'microsteps': 320, 'loss
↪': -22172234.0, 'train_time_minutes': 0.7826640844345093, 'throughput_average': 1744.
↪4691285659471, 'throughput_peak': 1745.4964663587539, 'batch_size': 8, 'max_length':␣
↪128}
Checkpointing...
Checkpointing done...
(hangs)
```

The fix is to add xm.rendezvous at the end of training to ensure all workers sync up before exiting the script dp_bert_large_pretrain_hdf5.py.

```
def _mp_fn(index, flags):
    torch.set_default_tensor_type('torch.FloatTensor')
    train_bert_hdf5(flags)
    xm.rendezvous("_mp_fn finished")
```

Reduced multi-node performance with Neuron PyTorch 1.12 (release 2.6)

Default BERT performance on multiple instances is reduced with Neuron PyTorch 1.12 (release 2.6). The workaround is to set the XLA flag XLA_TRANSFER_SEED_ASYNC=1.

8.4.6 Troubleshooting

The following are troubleshooting tips related to this tutorial. See *PyTorch Neuron on Trainium Troubleshooting Guide* for additional troubleshooting tips.

ModuleNotFoundError: No module named 'torch' , 'torch_xla', 'transformers', etc

If you encounter 'ModuleNotFoundError' messages while attempting to run the demo scripts, please ensure that you have activated the appropriate Python *virtualenv* which contains all of the demo dependencies:

```
cd ~
source <python virtual environment path>/bin/activate
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.5 Multi-Layer Perceptron Training Tutorial

MNIST is a standard dataset for handwritten digit recognition. A multi-layer perceptron (MLP) model can be trained with MNIST dataset to recognize hand-written digits. This tutorial starts with a 3-layer MLP training example in PyTorch on CPU, then show how to modify it to run on Trainium using PyTorch Neuron. It also shows how to do multiple worker data parallel MLP training.

Table of Contents

- *Setup environment and download examples*
- *Multi-layer perceptron MNIST model*
- *Single-worker MLP training script in PyTorch on CPU*
- *Single-worker MLP training on Trainium*
- *Multi-worker data-parallel MLP training using torchrun*
- *Known issues and limitations*

Note: Logs used in tutorials do not present latest performance numbers

For latest performance numbers visit *Neuron Performance*

8.5.1 Setup environment and download examples

Before running the tutorial please follow the installation instructions at:

- *Install PyTorch Neuron on Trn1*

Please set the storage of instance to *512GB* or more if you also want to run through the BERT pretraining and GPT pretraining tutorials.

For all the commands below, make sure you are in the virtual environment that you have created above before you run the commands:

```
source ~/aws_neuron_venv_pytorch/bin/activate
```

Install needed dependencies in your environment by running:

```
pip install pillow
```

Torchvision package is needed for MNIST dataset and has already been installed as part of *Install PyTorch Neuron on Trn1*. Installing Torchvision together with torch-neuronx ensures that the compatible version of Torchvision is selected. For example, torchvision==0.12 is compatible with torch==1.11 and torchvision==0.13 is compatible with torch==1.12.

To download the MNIST MLP examples, do:

```
git clone https://github.com/aws-neuron/aws-neuron-samples.git
cd aws-neuron-samples/torch-neuronx/training/mnist_mlp
```

8.5.2 Multi-layer perceptron MNIST model

In `model.py`, we define the multi-layer perceptron (MLP) MNIST model with 3 linear layers and ReLU activations, followed by a log-softmax layer. This model will be used in multiple example scripts.

8.5.3 Single-worker MLP training script in PyTorch on CPU

We will show how to modify a training script that runs on other platform to run on Trainium.

We begin with a single-worker MLP training script for running on the host CPUs of the Trainium instance. The training script imports the MLP model from `model.py`.

In this training script, we load the MNIST train dataset and, within the `main()` method, set the data loader to read batches of 32 training examples and corresponding labels.

Next we instantiate the MLP model and move it to the device. We use `device = 'cpu'` to illustrate the use of device in PyTorch. On GPU you would use `device = 'cuda'` instead.

We also instantiate the other two components of a neural network trainer: stochastic-gradient-descent (SGD) optimizer and negative-log-likelihood (NLL) loss function (also known as cross-entropy loss).

After the optimizer and loss function, we create a training loop to iterate over the training samples and labels, performing the following steps for each batch in each iteration:

- Zero gradients using:

```
optimizer.zero_grad()
```

- Move training samples and labels to device using the 'tensor.to' method.

- Perform forward/prediction pass using

```
output = model(train_x)
```

- The prediction results are compared against the corresponding labels using the loss function to compute the loss

```
loss_fn(output, train_label)
```

- The loss is propagated back through the model using chain-rule to compute the weight gradients

```
loss.backward()
```

- The weights are updated with a change that is proportional to the computed weights gradients

```
optimizer.step()
```

At the end of training we compute the throughput, display the final loss and save the checkpoint.

Expected CPU output:

```
----------Training ---------------
Train throughput (iter/sec): 286.96994718801335
Final loss is 0.1040
----------End Training ---------------
```

For a full tutorial on training in PyTorch, please see https://pytorch.org/tutorials/beginner/introyt/trainingyt.html.

Thus far we have used PyTorch without Trainium. Next, we will show how to change this script to run on Trainium.

8.5.4 Single-worker MLP training on Trainium

To run on Trainium, first we modify the CPU training script train_cpu.py to run with PyTorch Neuron torch_xla as described in *PyTorch Neuron for Trainium Getting Started Guide* by changing the device:

```
import torch_xla.core.xla_model as xm
device = xm.xla_device()
# or
device = 'xla'
```

When the model is moved to the XLA device using `model.to(device)` method, subsequent operations on the model are recorded for later execution. This is XLA's lazy execution which is different from PyTorch's eager execution. Within the training loop, we must mark the graph to be optimized and run on XLA device (NeuronCore) using xm.mark_step() (unless MpDeviceLoader is used as you will see in the next section). Without this mark, XLA cannot determine where the graph ends. The collected computational graph also gets compiled and executed when you request the value of a tensor such as by calling `loss.item()` or `print(loss)`.

To save a checkpoint, it is recommended to use the `xm.save()` function instead of `torch.save()` to ensure states are moved to CPU. `xm.save()` also prevents the "XRT memory handle not found" warning at the end of evaluation script (if the checkpoint saved using torch.save() is used for evaluation).

The resulting script `train.py` can be executed as `python3 train.py`. Again, note that we import the MLP model from `model.py`. When you examine the script, the comments that begin with 'XLA' indicate the changes required to make the script compatible with torch_xla.

Expected output on trn1.32xlarge (start from a fresh compilation cache, located at /var/tmp/neuron-compile-cache by default):

```
2022-04-12 16:15:00.000947: INFO ||NCC_WRAPPER||: No candidate found under /var/tmp/
↪neuron-compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_18200615679846498221.
2022-04-12 16:15:00.000949: INFO ||NCC_WRAPPER||: Cache dir for the neff: /var/tmp/
↪neuron-compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_18200615679846498221/
↪MODULE_0_SyncTensorsGraph.318_18200615679846498221_ip-172-31-69-14.ec2.internal-
↪8355221-28940-5dc775cd78aa2/83a0fd4a-b07e-4404-aa55-701ab3b2700c
........
Compiler status PASS
2022-04-12 16:18:05.000843: INFO ||NCC_WRAPPER||: Exiting with a successfully compiled␣
↪graph
2022-04-12 16:18:05.000957: INFO ||NCC_WRAPPER||: No candidate found under /var/tmp/
↪neuron-compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_5000680699473283909.
2022-04-12 16:18:05.000960: INFO ||NCC_WRAPPER||: Cache dir for the neff: /var/tmp/
↪neuron-compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_5000680699473283909/
↪MODULE_1_SyncTensorsGraph.390_5000680699473283909_ip-172-31-69-14.ec2.internal-8355221-
↪28940-5dc7767e5fc69/7d0a2955-11b4-42e6-b536-6f0f02cc68df
.
Compiler status PASS
2022-04-12 16:18:12.000912: INFO ||NCC_WRAPPER||: Exiting with a successfully compiled␣
↪graph
----------Training ---------------
Train throughput (iter/sec): 95.06756661972014
Final loss is 0.1979
----------End Training ---------------
```

If you re-run the training script a second time, you will see messages indicating that the compiled graphs are cached in the persistent cache from the previous run and that the startup time is quicker:

```
(aws_neuron_venv_pytorch_p36) [ec2-user@ip-172-31-69-14 mnist_mlp]$ python train.py |&␣
↪tee log_trainium
2022-04-12 16:21:58.000241: INFO ||NCC_WRAPPER||: Using a cached neff at /var/tmp/neuron-
↪compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_18200615679846498221/MODULE_0_
↪SyncTensorsGraph.318_18200615679846498221_ip-172-31-69-14.ec2.internal-8355221-28940-
↪5dc775cd78aa2/83a0fd4a-b07e-4404-aa55-701ab3b2700c/MODULE_0_SyncTensorsGraph.318_
↪18200615679846498221_ip-172-31-69-14.ec2.internal-8355221-28940-5dc775cd78aa2.neff.␣
↪Exiting with a successfully compiled graph
2022-04-12 16:21:58.000342: INFO ||NCC_WRAPPER||: Using a cached neff at /var/tmp/neuron-
↪compile-cache/USER_neuroncc-1.0.47218.0+162039557/MODULE_5000680699473283909/MODULE_1_
↪SyncTensorsGraph.390_5000680699473283909_ip-172-31-69-14.ec2.internal-8355221-28940-
↪5dc7767e5fc69/7d0a2955-11b4-42e6-b536-6f0f02cc68df/MODULE_1_SyncTensorsGraph.390_
↪5000680699473283909_ip-172-31-69-14.ec2.internal-8355221-28940-5dc7767e5fc69.neff.␣
↪Exiting with a successfully compiled graph
----------Training ---------------
Train throughput (iter/sec): 93.16748895384832
Final loss is 0.1979
----------End Training ---------------
```

Multiple graphs can be created during execution since there are differences between some iterations (first, steady state, last). After the first iteration, the graph for each iteration should remain the same from iteration to iteration. This allows XLA runtime to execute a previous compiled graph that has been cached in XLA runtime cache.

If the inner training loop has some control-flows, for example for gradient accumulation, the number of compiled graphs may increase due to the generation and consumption of intermediates as well as additional operations when the conditional path is taken.

8.5.5 Multi-worker data-parallel MLP training using torchrun

Data parallel training allows you to replicate your script across multiple workers, each worker processing a proportional portion of the dataset, in order to train faster.

The PyTorch distributed utility torchrun can be used to launch multiple processes in a server node for multi-worker data parallel training.

To run multiple workers in data parallel configuration using torchrun, modify the single-worker training script train.py as follows (below we use `xm` as alias for `torch_xla.core.xla_model` and `xmp` as alias for `torch_xla.distributed.xla_multiprocessing`):

1. Import XLA backend for torch.distributed using `import torch_xla.distributed.xla_backend`.
2. Use `torch.distributed.init_process_group('xla')` to initialize PyTorch XLA runtime and Neuron runtime.
3. Use XLA multiprocessing device loader (`MpDeviceLoader`) from `torch_xla.distributed` to wrap PyTorch data loader.
4. Use `xm.optimizer_step(optimizer)` to perform allreduce and take optimizer step.

XLA MpDeviceLoader is optimized for XLA and is recommended for best performance. It also takes care of marking the step for execution (compile and execute the lazily collected operations for an iteration) so no separate `xm.mark_step()` is needed.

The following are general best-practice changes needed to scale up the training:

1. Set the random seed to be the same across workers.
2. Scale up the learning rate by the number of workers. Use `xm.xrt_world_size()` to get the global number of workers.
3. Add distributed sampler to allow different worker to sample different portions of dataset.

Also, the `xm.save()` function used to save checkpoint automatically saves only for the rank-0 worker's parameters.

The resulting script is `train_torchrun.py` (note again that we import the MLP model from `model.py`):

Next we use the `torchrun` utility that is included with torch installation to run multiple processes, each using one NeuronCore. Use the option `nproc_per_node` to indicate the number of processes to launch. For example, to run on two NeuronCores on one Trn1 instance only, do:

```
torchrun --nproc_per_node=2 train_torchrun.py
```

NOTE: Currently we only support 1 and 2 worker configurations on trn1.2xlarge and 1, 2, 8, and 32-worker configurations on trn1.32xlarge.

Expected output on trn1.32xlarge (second run to avoid compilations):

```
----------Training ---------------
----------Training ---------------
... (Info messages truncated)
Train throughput (iter/sec): 163.25353269069706
Train throughput (iter/sec): 163.23261047441036
Final loss is 0.3469
Final loss is 0.1129
----------End Training ---------------
----------End Training ---------------
```

In another example, we run on two trn1.32xlarge instances launched with EFA-enabled interfaces, using EFA-enabled security group, and setup using *Install PyTorch Neuron on Trn1*. NOTE: To run on multiple instances, you will need to use trn1.32xlarge instances and using all 32 NeuronCores on each instance.

On the rank-0 Trn1 host (root), run with `--node_rank=0` using torchrun utility, and `--master_addr` set to rank-0 host's IP address:

```
export FI_EFA_USE_DEVICE_RDMA=1
export FI_PROVIDER=efa
torchrun --nproc_per_node=32 --nnodes=2 --node_rank=0 --master_addr=<root IP> --master_
↪port=2020 train_torchrun.py
```

On another Trn1 host, run with `--node_rank=1`, and `--master_addr` also set to rank-0 host's IP address:

```
export FI_EFA_USE_DEVICE_RDMA=1
export FI_PROVIDER=efa
torchrun --nproc_per_node=32 --nnodes=2 --node_rank=1 --master_addr=<root IP> --master_
↪port=2020 train_torchrun.py
```

It is important to launch rank-0 worker with `--node_rank=0` to avoid hang.

To train on multiple instances, it is recommended to use a ParallelCluster. For a ParallelCluster example, please see Train a model on AWS Trn1 ParallelCluster.

8.5.6 Known issues and limitations

MLP model is not optimized for performance. For the single-worker training, the performance can be improved by using MpDeviceLoader which exists in the multiprocessing example. For example, by setting `--nproc_per_node=1` in the torchrun example, you will see higher MLP performance.

```
(aws_neuron_venv_pytorch_p36) [ec2-user@ip-172-31-69-14 mnist_mlp]$ torchrun --nproc_per_
↪node=1 train_torchrun.py

----------Training ---------------
... (Info messages truncated)
Train throughput (iter/sec): 192.43508922834008
Final loss is 0.2720
----------End Training ---------------
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.6 PyTorch Neuron for Trainium Hugging Face BERT MRPC task fine-tuning using Hugging Face Trainer API

In this tutorial, we show how to run a Hugging Face script that uses Hugging Face Trainer API to do fine-tuning on Trainium. The example follows the text-classification example which fine-tunes BERT-base model for sequence classification on the GLUE benchmark.

Table of Contents

- *Setup and compilation*
- *Single-worker training*
- *Multi-worker training*
- *Converting BERT pretrained checkpoint to Hugging Face pretrained model format*
- *Known issues and limitations*

Note: Logs used in tutorials do not present latest performance numbers

For latest performance numbers visit *Neuron Performance*

8.6.1 Setup and compilation

Before running the tutorial please follow the installation instructions at:

- *Install PyTorch Neuron on Trn1*

Please set the storage of instance to *512GB* or more if you also want to run through the BERT pretraining and GPT pretraining tutorials.

For all the commands below, make sure you are in the virtual environment that you have created above before you run the commands:

```
source ~/aws_neuron_venv_pytorch/bin/activate
```

First we install recent version of HF transformers, scikit-learn and evaluate packages in our environment. In this example, we chose version 4.15.0:

```
pip install -U transformers==4.15.0 datasets evaluate scikit-learn
```

Next we download the transformers source, checking out the tag that matches the installed version:

```
cd ~/
git clone https://github.com/huggingface/transformers --branch v4.15.0
```

Go into the examples BERT training directory:

```
cd ~/transformers/examples/pytorch/text-classification
```

Edit the python script run_glue.py and add the following lines after the Python imports. They set the compiler flag for transformer model type and enable data parallel training using torchrun:

```
# Set compiler flag to compile for transformer model type
os.environ["NEURON_CC_FLAGS"] = os.environ.get('NEURON_CC_FLAGS', '') + " --model-
↪type=transformer"

# Enable torchrun
import torch
import torch_xla.distributed.xla_backend
if os.environ.get("WORLD_SIZE"):
    torch.distributed.init_process_group('xla')
```

(continues on next page)

(continued from previous page)

```
# Fixup to enable distributed training with XLA
from packaging import version
from transformers import __version__
if version.parse(__version__) < version.parse("4.20.0"):
    Trainer._wrap_model = lambda self, model, training=True: model
else:
    Trainer._wrap_model = lambda self, model, training=True, dataloader=None: model

# Workaround for NaNs seen with transformers version >= 4.21.0
# https://github.com/aws-neuron/aws-neuron-sdk/issues/593
if os.environ.get("XLA_USE_BF16") or os.environ.get("XLA_DOWNCAST_BF16"):
    transformers.modeling_utils.get_parameter_dtype = lambda x: torch.bfloat16
```

We will run MRPC task fine-tuning following the example in README.md. In this part of the tutorial we will use the Hugging Face model hub's pretrained `bert-large-uncased` model.

We precompile the model and training script to warm up the persistent graph cache (Neuron Cache) such that the actual run has fewer compilations (faster run time). We use full BF16 casting using XLA_USE_BF16=1 to enable best performance. Create a "compile" file with the following content, change it to executable, and run it:

```
export TASK_NAME=mrpc
neuron_parallel_compile XLA_USE_BF16=1 python3 ./run_glue.py \
--model_name_or_path bert-large-uncased \
--task_name $TASK_NAME \
--do_train \
--max_seq_length 128 \
--per_device_train_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 2 \
--overwrite_output_dir \
--output_dir /tmp/$TASK_NAME/ |& tee log_compile_train
```

Note that we separating out the train/eval steps during compilation. Also, we reduce batch size to 8 which is currently optimal for Trainium. Also, to reduce time to extract graph, we limit the trial run to 2 epochs. Please ignore the results from this trial run as it is only for extracting and compiling the XLA graphs.

8.6.2 Single-worker training

After precompilation, the actual run will be faster with minimal additional compilations. NOTE: currently the evaluation step would incur full compilation (see Known Issues below).

To run the actual training, create a "run" file with the following content and run it, again using full BF16 casting (XLA_USE_BF16=1) to enable best performance.:

```
export TASK_NAME=mrpc
XLA_USE_BF16=1 python3 ./run_glue.py \
--model_name_or_path bert-large-uncased \
--task_name $TASK_NAME \
--do_train \
--do_eval \
--max_seq_length 128 \
--per_device_train_batch_size 8 \
```

(continues on next page)

(continued from previous page)

```
--learning_rate 2e-5 \
--num_train_epochs 5 \
--overwrite_output_dir \
--output_dir /tmp/$TASK_NAME/ |& tee log_run
```

Rerunning the same script a second time would show quicker execution as the compiled graphs are already cached in persistent cache.

8.6.3 Multi-worker training

The above script would run one worker on one NeuronCore. To run on multiple cores, run the run_glue.py script with torchrun using `--nproc_per_node=N` option to specify the number of workers (N=2 for trn1.2xlarge, and N=2, 8, or 32 for trn1.32xlarge). The following example runs 2 workers:

```
export TASK_NAME=mrpc
XLA_USE_BF16=1 torchrun --nproc_per_node=2 ./run_glue.py \
--model_name_or_path bert-large-uncased \
--task_name $TASK_NAME \
--do_train \
--do_eval \
--max_seq_length 128 \
--per_device_train_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 5 \
--overwrite_output_dir \
--output_dir /tmp/$TASK_NAME/ |& tee log_run_2w
```

During run, you will notice that the "Total train batch size" is now 16 and the "Total optimization steps" is now half the number for one worker training.

8.6.4 Converting BERT pretrained checkpoint to Hugging Face pretrained model format

If you have a pretrained checkpoint (i.e., from the BERT phase 2 pretraining tutorial), you can run the script below (saved as "convert.py") to convert BERT pretrained saved checkpoint to Hugging Face pretrained model format. An example phase 2 pretrained checkpoint can be downloaded from `s3://neuron-s3/training_checkpoints/pytorch/dp_bert_large_hf_pretrain/ckpt_29688.pt`. Note that here we also use the `bert-large-uncased` model configuration to match the BERT-Large model trained following BERT phase 2 pretraining tutorial.

```
import os
import sys
import argparse
import torch
import transformers
from transformers import (
    BertForPreTraining,
)
import torch_xla.core.xla_model as xm
from transformers.utils import check_min_version
from transformers.utils.versions import require_version
```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--model_name', type=str, default='bert-large-uncased',  help=
↪"Path to model identifier from huggingface.co/models")
    parser.add_argument('--output_saved_model_path', type=str, default='./hf_saved_model
↪', help="Directory to save the HF pretrained model format.")
    parser.add_argument('--checkpoint_path', type=str, required=True, help="Path to␣
↪pretrained checkpoint which needs to be converted to a HF pretrained model format")
    args = parser.parse_args(sys.argv[1:])

    model = BertForPreTraining.from_pretrained(args.model_name)
    check_point = torch.load(args.checkpoint_path, map_location='cpu')
    model.load_state_dict(check_point['model'], strict=False)
    model.save_pretrained(args.output_saved_model_path, save_config=True, save_
↪function=xm.save)
```

Run the conversion script as:

```
python convert.py --checkpoint_path ckpt_29688.pt
```

After conversion, the new Hugging Face pretrained model is stored in the output directory specified by the `--output_saved_model_path` option which is `hf_saved_model` by default. You will use this directory in the next step.

To run the actual fine-tuning, create a "run_converted" file with the following script and run it:

```
export TASK_NAME=mrpc
XLA_USE_BF16=1 torchrun --nproc_per_node=2 ./run_glue.py \
--model_name_or_path hf_saved_model \
--tokenizer_name bert-large-uncased \
--task_name $TASK_NAME \
--do_train \
--do_eval \
--max_seq_length 128 \
--per_device_train_batch_size 8 \
--learning_rate 2e-5 \
--num_train_epochs 5 \
--overwrite_output_dir \
--output_dir /tmp/$TASK_NAME/ |& tee log_run_converted
```

8.6.5 Known issues and limitations

The following are currently known issues:

- Long compilation times: this can be alleviated with `neuron_parallel_compile` tool to extract graphs from a short trial run and compile them in parallel ahead of the actual run, as shown above.
- When precompiling using batch size of 16 on trn1.2xlarge, you will see `ERROR ||PARALLEL_COMPILE||: parallel compilation with neuronx-cc exited with error.Received error code: -9.` To workaround this error, please set NEURON_PARALLEL_COMPILE_MAX_RETRIES=1 in the environment.
- When running HuggingFace BERT (any size) fine-tuning tutorial or pretraining tutorial with transformers version >= 4.21.0 and using XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1, you will see NaNs in the loss

immediately at the first step. More details on the issue can be found at pytorch/xla#4152. The workaround is to use 4.20.0 or earlier or add `transformers.modeling_utils.get_parameter_dtype = lambda x: torch.bfloat16` to your Python script (i.e. run_glue.py).

- Using `neuron_parallel_compile` tool to run `run_glue.py` script with both train and evaluation options (`--do_train` and `--do_eval`), you will encounter INVALID_ARGUMENT error. To avoid this, only enable train for parallel compile (`--do_train`). This will cause compilations during evaluation step. The INVALID_ARGUMENT error will fixed in release 2.6 together with latest transformers package version 4.25.1.
- With release 2.6 and transformers==4.25.1, using `neuron_parallel_compile` tool to run `run_glue.py` script with both train and evaluation option (`--do_train` and `--do_eval`), you will encounter harmless error `ValueError: Target is multiclass but average='binary'`
- Reduced accuracy for RoBerta-Large is seen with Neuron PyTorch 1.12 (release 2.6) in FP32 mode with compiler BF16 autocast. The workaround is to set NEURON_CC_FLAGS="–auto-cast none" or set NEURON_RT_STOCHASTIC_ROUNDING_EN=1.
- Some recompilation is seen at the epoch boundary even after `neuron_parallel_compile` is used.
- When running multi-worker training, you may see the process getting killed at the time of model saving. This happens because the transformers `trainer.save_model` api uses `xm.save` for saving models. This api is known to cause high host memory usage in multi-worker setting see Saving and Loading XLA Tensors in . Coupled with a compilation at the same time results in a host OOM. To avoid this issue, we can: Precompile all the graphs in multi-worker training. This can be done by running the multi-worker training first with `neuron_parallel_compile` followed by the actual training. This would avoid the compilation at model save during actual training.

The following are currently known limitations:

- Only MRPC task fine-tuning is supported. Other tasks will be supported in the future.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.7 Megatron-LM GPT Pretraining Tutorial

GPT is a large language model that excels at many natural language processing (NLP) tasks. It is derived from the decoder part of the Transformer. Neuron Reference For Megatron-LM is a library that enables large-scale distributed training of language models such as GPT and is adapted from Megatron-LM. This tutorial explains how to run the Neuron reference for Megatron-LM GPT pretraining on Trainium.

The AWS Neuron SDK provides access to Trainium devices through an extension of PyTorch/XLA - a library that includes the familiar PyTorch interface along with XLA-specific additions. For Trainium customers, this means that existing PyTorch training scripts can be executed on Trn1 instances with minimal code modifications. For additional details relating to PyTorch/XLA, please refer to the official PyTorch/XLA documentation.

To run on Trainium, Neuron Reference For Megatron-LM library includes the following changes:

- GPU devices are replaced with Pytorch/XLA devices.
- Pytorch/XLA distributed backend is used to bridge the PyTorch distributed APIs to XLA communication semantics.
- Pytorch/XLA MpDeviceLoader is used for the data ingestion pipelines. Pytorch/XLA MpDeviceLoader helps improve performance by overlapping the three execution steps: tracing, compilation and data batch loading to the device.
- CUDA APIs are mapped to generic PyTorch APIs.

- CUDA fused optimizers are replaced with generic PyTorch alternatives.

The GPT example in this tutorial is an adaptation of the original Megatron-LM GPT example, trained using the Wikipedia dataset.

Table of Contents

- *Install PyTorch Neuron*
- *Download Preprocessed Wikipedia Dataset*
- *Setting up the training environment on trn1.32xlarge*
- *GPT Pretraining Python Script*
- *GPT Training Shell Script*
- *Initiating a Training Job*
- *Monitoring Training Job Progress*
- *Monitoring Training Job Progress using neuron-top*
- *Monitoring Training Job Progress using TensorBoard*
- *Finishing the tutorial*
- *Running a multi-node GPT*
- *Checkpointing GPT Model*
- *Preparing Wikipedia Dataset from Scratch*
- *Known issues and limitations*
  - *No broadcast support*
  - *No pipeline parallel support*
  - *Dropout is disabled*
  - *Error: cannot import name 'helpers' from 'megatron.data'*
  - *Error: Out of space while checkpointing*
- *Troubleshooting*

Note: Logs used in tutorials do not present latest performance numbers

For latest performance numbers visit *Neuron Performance*

8.7.1 Install PyTorch Neuron

Before running the tutorial please follow the installation instructions at:

- *Install PyTorch Neuron on Trn1*

Please set the storage of instance to *512GB* or more if you intent to run multiple experiments and save many checkpoints.

8.7.2 Download Preprocessed Wikipedia Dataset

Download the vocabulary file, the merge table file, and the preprocessed Wikipedia dataset using the following commands:

```
export DATA_DIR=~/examples_datasets/gpt2

mkdir -p ${DATA_DIR} && cd ${DATA_DIR}

wget https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-vocab.json
wget https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-merges.txt
aws s3 cp s3://neuron-s3/training_datasets/gpt/wikipedia/my-gpt2_text_document.bin .  --
↪no-sign-request
aws s3 cp s3://neuron-s3/training_datasets/gpt/wikipedia/my-gpt2_text_document.idx .  --
↪no-sign-request
aws s3 cp s3://neuron-s3/training_datasets/gpt/wikipedia/license.txt .  --no-sign-request
```

See section `Preparing Wikipedia dataset from scratch` if you would like to recreate the preprocessed dataset from scratch.

8.7.3 Setting up the training environment on trn1.32xlarge

Please follow the instructions to setup Python virtual environment with Neuron packages.

Install Python3 development package needed to build the data helpers tools. If you are on Amazon Linux, do:

```
sudo yum install -y python3-devel
```

If you are on Ubuntu, do:

```
sudo apt install -y python3-dev
```

Clone the AWS Neuron Reference for Megatron-LM package, install dependencies, and build the data helpers tool:

```
cd ~/
git clone https://github.com/aws-neuron/aws-neuron-reference-for-megatron-lm.git
pip install pybind11 regex
pushd .
cd aws-neuron-reference-for-megatron-lm/megatron/data/
make
popd
```

8.7.4 GPT Pretraining Python Script

The GPT pretraining python script is a wrapper that imports the Megatron-LM library modules and sets up the pieces needed by the Megatron-LM trainer: GPT model, loss function, forward pass, data provider. It is adapted from pretrain_gpt.py. The Neuron changes are:

- Use XLA device
- Not using mpu.broadcast_data as it is currently unsupported. Instead each worker reads the data in parallel.
- Use int instead of long datatype for token data

The script is available at `~/aws-neuron-reference-for-megatron-lm/pretrain_gpt.py`

8.7.5 GPT Training Shell Script

The GPT training shell script runs the above python script with following model configurations (for 6.7 billion parameters model):

- Number of layers: 32
- Hidden size: 4096
- Number attention heads: 32
- Sequence length: 2048
- Max positional embeddings size: 2048

The following training parameters are used:

- The number of gradient accumulation microsteps is 64, with worker batch size of 1.
- The tensor parallelism degree is 8.
- The data parallelism degree is 4.
- The number of workers is 32.

Additionally, the script uses:

- CPU intitialization
- AdamW optimizer (default).
- Gradient clipping.
- No CUDA fusions (bias-gelu, masked-softmax, bias-dropout)
- Disabled contiguous buffer in local DDP
- Option `--distributed-backend xla` picks the XLA distributed backend to bridge the Pytorch distributed APIs to XLA communication semantics.

See this link for a full list of options and their descriptions.

Note: Not all options are supported. Currently only tensor-parallel and data-parallel modes in Neuron Reference For Megatron-LM are supported. We support tensor-parallel degree of 8 and data-parallel degree of upto 64.

The script for running on a single node is available at `~/aws-neuron-reference-for-megatron-lm/examples/pretrain_gpt3_6.7B_32layers_bf16.sh`

This shell script expects dataset files to be located in ~/examples_datasets/gpt2/ following the steps above. If you place the dataset files in another location, please update the DATA_PATH variable in the shell script.

8.7.6 Initiating a Training Job

To run the GPT example, first activate the Python virtual environment and change to the Megatron-LM package location:

```
source ~/aws_neuron_venv_pytorch/bin/activate
cd ~/aws-neuron-reference-for-megatron-lm/
```

Next, run the parallel compilations of graphs in order to reduce compilation time during the actual run.

```
neuron_parallel_compile sh ./examples/pretrain_gpt3_6.7B_32layers_bf16.sh
```

This command performs a short trial run of the training script to extract graphs and then do parallel compilations on those graphs before populating the persistent cache with compiled graphs. This helps reduce the compilation time during the actual run of the training script.

Note: Please ignore the results of the trial run as they are not the actual execution results.

If some or all the graphs were already compiled and cached in the persistent cache, then fewer or none of the graphs would need compilation. To force recompilation, you can remove the cache directory at `/var/tmp/neuron-compile-cache/`.

Compilation is recommended if there are some changes in the script (such as batch size, number of layers, number of workers, etc.). Compilation will only happen if the model graph or its parameters/compilation flags change.

Finally, run the script for the actual run:

```
sh ./examples/pretrain_gpt3_6.7B_32layers_bf16.sh
```

During the run, you will see outputs like below, some lines showing throughput and loss statistics every global step.

```
`iteration     4873/   10000 | consumed samples:       311872 | elapsed time per␣
↪iteration (ms): 8718.9 | learning rate: 1.500E-04 | global batch size:    64 | lm loss:
↪ 3.296875E+00 | grad norm: 0.430 | throughput: 7.340`
```

8.7.7 Monitoring Training Job Progress

Using a single Trn1 instance with 32 NeuronCores, the current GPT pretraining will run for ~81 hours. During this time, you will see the average loss metric begin at 11 and ultimately converge to ~3.2. Throughput for the training job will be ~7.3 seq/sec.

8.7.8 Monitoring Training Job Progress using neuron-top

With the training job still running, launch a second SSH connection into the trn1 instance, and use the `neuron-top` command to examine the aggregate NeuronCore utilization.

8.7.9 Monitoring Training Job Progress using TensorBoard

The demo includes TensorBoard-compatible logging, which allows the learning rate and training metrics to be monitored in real-time. By default, the training script logs metrics to the following TensorBoard log directory `~/aws-neuron-reference-for-megatron-lm/tb_*`.

In order to view your training metrics in TensorBoard, first run the following commands in your SSH session:

```
source ~/aws_neuron_venv_pytorch/bin/activate
cd ~/aws-neuron-reference-for-megatron-lm/
tensorboard --logdir ./
```

Once running, open a new SSH connection to the instance and port-forward TCP port 6006 (ex: -L 6006:127.0.0.1:6006). Once the tunnel is established, TensorBoard can then be accessed via web browser at the following URL: http://localhost:6006. Please note that you will not be able to access TensorBoard if you disconnect your port-forwarding SSH session to the Trainium instance.

8.7.10 Finishing the tutorial

Once you are ready, and the training throughput is as expected, there are a couple of options for finishing the GPT pretraining demo:

Allow the training script to run to completion. If you would like to observe the training script run to completion, it is recommended to launch the training script from a terminal multiplexer such as `tmux` or `screen`, and then detach the session so that the training script can run in the background. With this approach, you can safely let the training script run unattended, without risk of an SSH disconnection causing the training job to stop running.

Stop the training job early. To stop the training job early, press CTRL-C in the terminal window in which you launched the training script. In some cases, if you manually cancel a job using CTRL-C and then later want to run the job again, you might first need to terminate all the python processes by the command `killall -9 python3` .

8.7.11 Running a multi-node GPT

We use SLURM to launch multi-node GPT training jobs. Like single node runs, we have a precompilation step followed by the actual run. To precompile:

```
sbatch examples/pretrain_gpt3_6.7B_compile.slurm
```

This will precompile the script `examples/pretrain_gpt3_6.7B_32layers_bf16_bs1024_slurm.sh` on all the nodes and populate the caches.

To run the compiled model:

```
sbatch examples/pretrain_gpt3_6.7B.slurm
```

The number of nodes is currently set to 16 and since the tensor-parallel degree used is 8, the data-parallel degree is automatically computed to be 64, resulting in a 8x64 two dimensional mesh parallelism.

The tensorboard logs are written by the last rank and will be in the TensorBoard log directory `~/aws-neuron-reference-for-megatron-lm/tb_*`.

Compared to the single-node script, we use an increased batch size of 1024 which gives us a throughput bump of ~98 seq/sec. The number of iterations is also increased with changes in the hyperparameters pertaining to learning rates, weight decay.

8.7.12 Checkpointing GPT Model

A new mode of checkpointing using serialized tensor and staggered save/load is supported to alleviate memory pressure. To save the model, add the lines:

```
--save-xser $CHECKPOINT_PATH
--save-interval 1500
```

This will save the checkpoint at path variable provided for every 1500 iterations.

Note: Please note that the model saves all the model weights, optimizer and rng states (~76GB for a 32 layermodel). And if checkpointed frequently can quickly lead to low disk storage. Make sure there is enough disk space.

To load the checkpoint, we first need to remove `--use-cpu-initialization` from the script and then add

```
--load-xser $CHECKPOINT_PATH
```

Note: Please note not removing the –use-cpu-initialization flag may lead to out-of-memory execution and result in unstable resumption of training.

8.7.13 Preparing Wikipedia Dataset from Scratch

The process of preparing the Wikipedia dataset follows the original Megatron-LM documentation. You will need a large c5 machine like c5n.18xlarge and using the latest Deep Learning AMI. First download the Wikipedia dataset. Depending on the network bandwidth, this is expected to be about ~65 minutes.

```
export WIKI_DIR=~/examples_datasets/wiki
mkdir -p $WIKI_DIR && cd $WIKI_DIR

wget https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
```

Download the vocabulary and merge table files for the desired model. This example uses the GPT-2 model:

```
export DATA_DIR=~/examples_datasets/gpt2
export GPT2_DATA=${DATA_DIR}/gpt2

mkdir -p ${GPT2_DATA} && cd ${GPT2_DATA}

wget https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-vocab.json
wget https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-merges.txt

mkdir -p ${GPT2_DATA}/checkpoint
wget --content-disposition https://api.ngc.nvidia.com/v2/models/nvidia/megatron_lm_345m/
↪versions/v0.0/zip -O ${GPT2_DATA}/checkpoint/megatron_lm_345m_v0.0.zip
```

Extract the downloaded data using WikiExtractor (this step takes about 2 hours):

```
git clone https://github.com/attardi/wikiextractor.git /tmp/wikiextractor
cd /tmp/wikiextractor
python -m wikiextractor.WikiExtractor --json ~/examples_datasets/wiki/enwiki-latest-
↪pages-articles.xml.bz2 --output ~/examples_datasets/wiki/text/ -q --processes 70 2>&1␣
↪| tee wikiextract.out &
```

(continues on next page)

(continued from previous page)

The Wikiextractor first preprocesses the template of all pages sequentially, followed by a Map/Reduce process for extracting the pages and converting to the loose json format required by Megatron-LM.

Once the extraction completes, we merge the text files with (~2 minutes):

```
conda activate pytorch_latest_p37
cd ~/examples_datasets/wiki
find ~/examples_datasets/wiki/text/ -name wiki* | parallel -m -j 70 "cat {} >>␣
↪mergedfile.json"
```

The `mergedfile.json` size on disk is 16GB. With it, create the binary data format for Megatron GPT2. NOTE: Refer to this solution if an `IndexError: list index out of range` occurs. To create the binary data, type the following command:

```
python ~/aws-neuron-reference-for-megatron-lm/tools/preprocess_data.py \
    --input ~/examples_datasets/wiki/mergedfile.json \
    --output-prefix my-gpt2 \
    --vocab ~/examples_datasets/gpt2/gpt2-vocab.json \
    --dataset-impl mmap \
    --tokenizer-type GPT2BPETokenizer \
    --merge-file ~/examples_datasets/gpt2/gpt2-merges.txt \
    --append-eod \
    --workers 70
```

Files my-gpt2_text_document.* are generated after about 12 minutes.

8.7.14 Known issues and limitations

No broadcast support

Currently, the mpu.broadcast_data is unsupported on Trainium.

No pipeline parallel support

Currently, only tensor parallel and data parallel are supported and there is no pipeline parallel support in Neuron Reference For Megatron-LM.

Dropout is disabled

Currently, dropout is disabled in the example.

Error: cannot import name 'helpers' from 'megatron.data'

You may encounter the error "cannot import name 'helpers' from 'megatron.data'" like below:

```
Exception in device=NEURONT:0: cannot import name 'helpers' from 'megatron.data' (/home/
↪ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/__init__.py)
Traceback (most recent call last):
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37/lib64/python3.7/site-packages/torch_
↪xla/distributed/xla_multiprocessing.py", line 373, in _mp_start_fn
    _start_fn(index, pf_cfg, fn, args)
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37/lib64/python3.7/site-packages/torch_
↪xla/distributed/xla_multiprocessing.py", line 367, in _start_fn
    fn(gindex, *args)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/pretrain_gpt_mp.py", line␣
↪138, in pretrain_mp
    forward_step, args_defaults={'tokenizer_type': 'GPT2BPETokenizer'})
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/training.py", line␣
↪162, in pretrain
    train_valid_test_dataset_provider)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/training.py", line␣
↪1021, in build_train_valid_test_data_iterators
    train_val_test_num_samples)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/pretrain_gpt_mp.py", line␣
↪128, in train_valid_test_datasets_provider
    skip_warmup=(not args.mmap_warmup))
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/gpt_dataset.py
↪", line 43, in build_train_valid_test_datasets
    seq_length, seed, skip_warmup)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/gpt_dataset.py
↪", line 118, in _build_train_valid_test_datasets
    train_dataset = build_dataset(0, 'train')
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/gpt_dataset.py
↪", line 115, in build_dataset
    seq_length, seed)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/gpt_dataset.py
↪", line 156, in __init__
    num_samples, seq_length, seed)
  File "/home/ec2-user/aws-neuron-reference-for-megatron-lm/megatron/data/gpt_dataset.py
↪", line 274, in _build_index_mappings
    from megatron.data import helpers
ImportError: cannot import name 'helpers' from 'megatron.data' (/home/ec2-user/aws-
↪neuron-reference-for-megatron-lm/megatron/data/__init__.py)
```

To fix this, please go into aws-neuron-reference-for-megatron-lm/megatron/data/ and do "make":

```
pip install pybind11
pushd .
cd aws-neuron-reference-for-megatron-lm/megatron/data/
make
popd
```

Error: Out of space while checkpointing

You may seem an error as follows. The model checkpoints are large as they dump all the model weights, optimizer and rng states. And if these are frequently checkpointed, the storage can run out fast. Please make sure you have enough disk space.

```
Traceback (most recent call last):
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37/lib64/python3.7/site-packages/torch/
↪serialization.py", line 380, in save
    _save(obj, opened_zipfile, pickle_module, pickle_protocol)
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37/lib64/python3.7/site-packages/torch/
↪serialization.py", line 604, in _save
    zip_file.write_record(name, storage.data_ptr(), num_bytes)
OSError: [Errno 28] No space left on device
```

8.7.15 Troubleshooting

See *PyTorch Neuron (torch-neuronx) for Training Troubleshooting Guide*

This document is relevant for: `Trn1`

Note: To use Jupyter Notebook see:

- setup-jupyter-notebook-steps-troubleshooting
- running-jupyter-notebook-as-script

Additional Examples (`torch-neuronx`)

- AWS Neuron Reference for Megatron-LM GitHub Repository
- AWS Neuron Samples for AWS ParallelCluster
- AWS Neuron Samples GitHub Repository

API Reference Guide (`torch-neuronx`)

This document is relevant for: `Trn1`

8.8 PyTorch Neuron neuron_parallel_compile CLI (`torch-neuronx`)

PyTorch Neuron performs just-in-time compilation of graphs during execution. At every step, a graph is traced. If the traced graph varies from the previous executions, it is compiled by the neuron compiler. For large models, the compilation time for each graph can be high. Moreover, because of JIT, we would compile all these graphs sequentially, hence incurring huge compilation penalty.

To reduce this compilation time during execution, the `neuron_parallel_compile` utility is provided as part of PyTorch Neuron installation. The `neuron_parallel_compile` will extract graphs from a trial run of your script, perform parallel pre-compilation of the graphs, and populate the Neuron Cache on disk with compiled graphs. Your trial run should be limited to a few steps (eg.10-15), enough for the utility to extract the different graphs needed for full execution. To run the utility:

```
neuron_parallel_compile <run commands>
```

Where `<run commands>` are the commands to run a short run (i.e. 10 steps) to trace training loops for pre-compilation. The example for the run command is `torchrun --nproc_per_node=2 <train script>`, where train script accepts `--steps_this_run` option to limit number of run steps:

```
neuron_parallel_compile torchrun --nproc_per_node=2 <train script> --steps_this_run=10
```

NOTE: To avoid hang during `neuron_parallel_compile` run, please make sure to use xm.save instead of torch.save to save checkpoints.

You may notice that the output from the model is invalid when you use `neuron_parallel_compile`. This is because, when you initiate your training run command with `neuron_parallel_compile` , the utility will run your command with certrain env variables that would put your training script into graph extraction mode. In this mode, no real execution is performed and the outputs are invalid.

Once the `neuron_parallel_compile` finishes compilation of all graphs, it will copy all the compilation results into the Neuron Cache.

Now if you run your script (without `neuron_parallel_compile`), it will be faster since the compiled graphs are already cached.

```
torchrun --nproc_per_node=2 <train script>
```

`Note`: Except for the option to limit number of run steps (such as `--steps_this_run`), the other options of `<run commands>` must match between the pre-compilation and actual run. If this is not the case, you may see additional compilations during training run because of new graphs getting generated, resulting in cache miss.

There may be additional compilations due to unreached execution paths (in case the execution path is not reached in the first few steps of graph extraction), or changes in parameters such as number of data parallel workers.

Two additional utility flags are provided:

`NEURON_PARALLEL_COMPILE_MAX_RETRIES` :

- Set the maximum number of retries when using `neuron_parallel_compile` tool. If set to N, the tool will try compilation N more time(s) if the first graph compilation failed. Example: Set NEURON_PARALLEL_COMPILE_MAX_RETRIES=1 when precompiling on trn1.2xlarge where there's limited host memory and CPU resources. Default is 0.

`NEURON_IGNORE_TRAINING_SCRIPT_ERROR_AND_COMPILE` :

- When using neuron_parallel_compile, if you want to ignore the error in training script and compile the accumulated HLO graphs, you can do so by setting this environment variable. Example: If NEURON_IGNORE_TRAINING_SCRIPT_ERROR_AND_COMPILE=1 is set when using `neuron_parallel_compile`, a crash in the training script would be ignored and the graphs collected upto the crash would be compiled.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.9 PyTorch Neuron Environment Variables (torch-neuronx)

Environment variables allow modifications to PyTorch Neuron behavior without requiring code change to user script. It is recommended to set them in code or just before invoking the python process, such as NEURON_FRAMEWORK_DEBUG=1 python3 <script> to avoid inadvertently changing behavior for other scripts. Environment variables specific to PyTorch Neuron are (experimental ones are noted):

NEURON_CC_FLAGS:

- Compiler options. Full compiler options are described in the mixed-precision-casting-options. Additional options for the persistent cache can be found in the "Persistent cache" section below.

NEURON_FRAMEWORK_DEBUG **[Experimental]**:

- Enable dumping of XLA graphs in both HLO format (intermediate representation) and text form for debugging.

NEURON_EXTRACT_GRAPHS_ONLY **[Experimental]**:

- Dump the XLA graphs in HLO format (intermediate representation) and execute empty stubs with zero outputs in order to allow multiple XLA graphs to be traced through a trial execution. Used for ahead-of-time graph extraction for parallel compilation in neuron_parallel_compile tool. This environment variable can be checked in the training script to prevent checking of bad outputs during trial run.

NEURON_NUM_RECENT_MODELS_TO_KEEP **[Experimental]**:

- Keep only N number of graphs loaded in Neuron runtime for each process, where N is the value this environment variable is set to. Default is to keep all graphs loaded by a process.

NEURON_PARALLEL_COMPILE_MAX_RETRIES **[Experimental]**:

- Set the maximum number of retries when using neuron_parallel_compile tool. If set to N, the tool will try compilation N more time(s) if the first graph compilation failed. Example: Set NEURON_PARALLEL_COMPILE_MAX_RETRIES=1 when precompiling on trn1.2xlarge where there's limited host memory and CPU resources. Default is 0.

NEURON_IGNORE_TRAINING_SCRIPT_ERROR_AND_COMPILE **[Experimental]**:

- When using neuron_parallel_compile, if you want to ignore the error in training script and compile the accumulated HLO graphs, you can do so by setting this environment variable. Example: If NEURON_IGNORE_TRAINING_SCRIPT_ERROR_AND_COMPILE=1 is set when using neuron_parallel_compile, a crash in the training script would be ignored and the graphs collected upto the crash would be compiled.

NEURON_DUMP_HLO_SNAPSHOT **[Experimental]**:

- Dump the inputs, outputs, and graph in HLO format of a graph execution in a snapshot file. This variable can be set to 1, ON_NRT_ERROR, ON_NRT_ERROR_CPU, ON_NRT_ERROR_HYBRID to dump snapshots at every iteration using CPU memory, or dump only on errors automatically using device, host, and both device and host memory respectively.

NEURON_NC0_ONLY_SNAPSHOT **[Experimental]**:

- Dump only the snapshot associated with Neuron Core 0 when NEURON_NC0_ONLY_SNAPSHOT=1 and the NEURON_DUMP_HLO_SNAPSHOT flag is set.

BUCKET_CAP_MB **[PyTorch XLA]**:

- If there are many parameters, such as in BERT training, small allreduce sizes can limit performance. To improve performance, you can try increasing the bucket size using BUCKET_CAP_MB environment variable, which is set to 50MB by default. For example, BERT pretraining on multiple instances can see improved performance with BUCKET_CAP_MB=512.

XLA_USE_BF16 **[PyTorch XLA]**:

- When `XLA_USE_BF16=1`, PyTorch Neuron will automatically map both torch.float and torch.double tensors to bfloat16 tensors and turn on Stochastic Rounding mode. This can both reduce memory footprint and improve performance. Example: to enable bfloat16 autocasting and stochastic rounding, set XLA_USE_BF16=1 only, as stochastic rounding mode is on by default when XLA_USE_BF16=1. If you would like to preserve some tensors in float32, see `XLA_DOWNCAST_BF16` below.

`XLA_DOWNCAST_BF16` **[PyTorch XLA]**:

- When `XLA_DOWNCAST_BF16=1`, PyTorch Neuron will automatically map torch.float tensors to bfloat16 tensors, torch.double tensors to float32 tensors and turn on Stochastic Rounding mode. This can both reduce memory footprint and improve performance, while preserving some tensors in float32. Example: to enable float to bfloat16 and double to float autocasting and stochastic rounding, set XLA_DOWNCAST_BF16=1 only, as stochastic rounding mode is on by default when XLA_DOWNCAST_BF16=1. If you want to cast both torch.float and torch.double to bfloat16, please see `XLA_USE_BF16` above.

`NEURON_RT_STOCHASTIC_ROUNDING_EN` **[Neuron Runtime]**:

- When `NEURON_RT_STOCHASTIC_ROUNDING_EN=1`, PyTorch Neuron will use stochastic rounding instead of round-nearest-even for all internal rounding operations when casting from FP32 to a reduced precision data type (FP16, BF16, FP8, TF32). This feature has been shown to improve training convergence for reduced precision training jobs, such as when bfloat16 autocasting is enabled. This is set to 1 by default by PyTorch Neuron when XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1. To switch to round-nearest-even mode, please set `NEURON_RT_STOCHASTIC_ROUNDING_EN=0`.

`NEURON_RT_STOCHASTIC_ROUNDING_SEED` **[Neuron Runtime]**:

- Sets the seed for the random number generator used in stochastic rounding (see previous section). If this environment variable is not set, the seed is set to 0 by default. Please set `NEURON_RT_STOCHASTIC_ROUNDING_SEED` to a fixed value to ensure reproducibility between runs.

`NEURON_RT_VISIBLE_CORES` **[Neuron Runtime]**:

Integer range of specific NeuronCores needed by the process (for example, 0-3 specifies NeuronCores 0, 1, 2, and 3). You this environment variable when using torchrun to limit the launched processs to specific consecutive NeuronCores. To ensure best performance, the multi-core jobs requiring N NeuronCores for collective communication must be placed at the NeuronCore ID that starts at a multiple of N, where N is the world size limited to 1, 2, 8, 32. For example, a process using 2 NeuronCores can be mapped to 2 free NeuronCores starting at NeuronCore id 0, 2, 4, 6, etc, and a process using 8 NeuronCores can be mapped to 8 free NeuronCores starting at NeuronCore id 0, 8, 16, 24.

Additional Neuron runtime environment variables are described in runtime configuration documentation.

Additional XLA runtime environment variables are described in PyTorch-XLA troubleshooting guide.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.10 PyTorch Neuron (`torch-neuronx`) Profiling API

The profiler provides a method to generate a context manager to capture trace events at the operator or runtime level.

`torch_neuronx.experimental.profiler.`**`profile`**(*port=9012*, *ms_duration=60000*, *neuron_tensorboard_plugin_dir='logs/plugins/neuron'*, *profile_type='operator'*, *auto_start=True*, *delete_working=True*)

The *`torch_neuronx.experimental.profiler.profile()`* method retuns a `profile` context manager object. This object doesn't need to be used directly, as default options are set to auto capture events based on the `profile_type`.

The context manager will wrap around the entire model and training/inference loop. The context-manager is backwards-compatible with the torch_xla.debug.profiler``

Required Arguments

None

Optional Keyword Arguments

Keyword Arguments

- **port** (*int*) – Port to run the profiling GRPC server on. Default is 9012.
- **ms_duration** (*int*) – This defines how long the profiler will capture the HLO artifacts from the model to view in the profiler. The unit is in milliseconds. The default value is 60000 ms, or 1 minute.
- **neuron_tensorboard_plugin_dir** (*str*) – The directory the neuron tensorboard plugin will file write to. This will be `logs/plugins/neuron` by default/
- **profile_type** (*str*) – There is "trace" and "operator". "trace" is the Torch Runtime Trace Level, while "operator" is the Model Operator Trace Level. Default is "operator"
- **auto_start** (*bool*) – If set to true, the profiler will start profiling immediately. If set to false, the profiler can be set to start at a later condition. Refer to `profile.start()` for more details. Default is `True`.
- **delete_working** (*bool*) – If set to False turns off the deletion of temporary files. Default True.

Returns The traced *profile*

Return type ~profile

`torch_neuronx.experimental.profiler.profile.start()`

The *torch_neuronx.experimental.profiler.profile.start()* method starts the profiler if not started (i.e when `auto_start=False`). This function does not take in any parameters, nor return anything.

Required Arguments

None

Optional Keyword Arguments

None

Returns None

This document is relevant for: `Trn1`

Developer Guide (torch-neuronx)

This document is relevant for: `Trn1`

8.11 Developer Guide for Training with PyTorch Neuron (`torch-neuronx`)

Table of Contents

- *PyTorch Neuron*
  - *Neuron XLA device*
- *PyTorch Neuron single-worker training quick-start*
- *PyTorch Neuron multi-worker data parallel training using torchrun*
- *Conversion from Distributed Data Parallel (DDP) application*
- *PyTorch Neuron environment variables*
- *Persistent cache for compiled graphs*
- *Number of graphs*
- *Automatic casting of float tensors to BFloat16*
- *Automatic Mixed-Precision*
  - *BF16 mixed-precision using PyTorch Autocast*
- *Tips and Best Practices*
  - *Understand the lazy mode in PyTorch Neuron*
  - *Minimize the number of compilation-and-executions*
  - *Ensure common initial weights across workers*
  - *Use PyTorch/XLA's model save function*
- *FAQ*
  - *What is the difference between Trainium and Inferentia?*
- *Debugging and troubleshooting*

Trainium is designed to speed up model training and reduce training cost. It is available on the Trn1 instances. Each Trainium accelerator has two NeuronCores, which are the main neural network compute units.

PyTorch Neuron enables PyTorch users to train their models on Trainium's NeuronCores with little code change to their training code. It is based on the PyTorch/XLA software package.

This guide helps you get started with single-worker training and distributed training using PyTorch Neuron.

8.11.1 PyTorch Neuron

Neuron XLA device

With PyTorch Neuron the default XLA device is mapped to a NeuronCore. By default, one NeuronCore is configured. To use Neuron XLA device, specify the device as `xm.xla_device()` or `'xla'`:

```
import torch_xla.core.xla_model as xm
device = xm.xla_device()
```

or

```
device = 'xla'
```

PyTorch models and tensors can be mapped to the device as usual:

```
model.to(device)
tensor.to(device)
```

To move tensor back to CPU, do :

```
tensor.cpu()
```

or

```
tensor.to('cpu')
```

8.11.2 PyTorch Neuron single-worker training quick-start

PyTorch Neuron uses XLA to enable conversion of PyTorch operations to Trainium instructions. To get started on PyTorch Neuron, first modify your *training script* to use XLA in the same manner as described in PyTorch/XLA documentation and use XLA device:

```
import torch_xla.core.xla_model as xm

device = xm.xla_device()
# or
device = 'xla'
```

The NeuronCore is mapped to an XLA device. On Trainium instance, the XLA device is automatically mapped to the first available NeuronCore.

By default the above steps will enable the training script to run on one NeuronCore. NOTE: Each process is mapped to one NeuronCore.

Finally, add `mark_step` at the end of the training step to compile and execute the training step:

```
xm.mark_step()
```

These changes can be placed in control-flows in order to keep the script the same between PyTorch Neuron and CPU/GPU. For example, you can use an environment variable to disable XLA which would cause the script to run in PyTorch native mode (using CPU on Trainium instances and GPU on GPU instances):

```
device = 'cpu'
if not os.environ.get("DISABLE_XLA", None):
    device = 'xla'

...

    # end of training step
    if not os.environ.get("DISABLE_XLA", None):
        xm.mark_step()
```

More on the need for mark_step is at *Understand the lazy mode in PyTorch Neuron*.

For a full runnable example, please see the Single-worker MLP training on Trainium tutorial.

8.11.3 PyTorch Neuron multi-worker data parallel training using torchrun

Data parallel training allows you to replicate your script across multiple workers, each worker processing a proportional portion of the dataset, in order to train faster.

To run multiple workers in data parallel configuration, with each worker using one NeuronCore, first add additional imports for parallel dataloader and multi-processing utilities:

```
import torch_xla.distributed.parallel_loader as pl
```

Next we initialize the Neuron distributed context using the XLA backend for torch.distributed:

```
import torch_xla.distributed.xla_backend
torch.distributed.init_process_group('xla')
```

Next, replace `optimizer.step()` function call with `xm.optimizer_step(optimizer)` which adds gradient synchronization across workers before taking the optimizer step:

```
xm.optimizer_step(optimizer)
```

If you're using a distributed dataloader, wrap your dataloader in the PyTorch/XLA's `MpDeviceLoader` class which provides buffering to hide CPU to device data load latency:

```
parallel_loader = pl.MpDeviceLoader(dataloader, device)
```

Within the training code, use xm.xrt_world_size() to get the world size, and xm.get_ordinal to get the global rank of the current process.

Then run use PyTorch torchrun utility to run the script. For example, to run 32 worker data parallel training:

```
torchrun --nproc_per_node=32 <script and options>
```

To run on multiple instances, make sure to use trn1.32xlarge instances and use all 32 NeuronCores on each instance. For example, with two instances, on the rank-0 Trn1 host, run with –node_rank=0 using torchrun utility:

```
torchrun --nproc_per_node=32 --nnodes=2 --node_rank=0 --master_addr=<root IP> --master_
↪port=<root port> <script and options>
```

On another Trn1 host, run with –node_rank=1 :

```
torchrun --nproc_per_node=32 --nnodes=2 --node_rank=1 --master_addr=<root IP> --master_
↪port=<root port> <script and options>
```

It is important to launch rank-0 worker with –node_rank=0 to avoid hang.

More information about torchrun can be found PyTorch documentation at https://pytorch.org/docs/stable/elastic/run.html#launcher-api .

See the Multi-worker data-parallel MLP training using torchrun tutorial for a full example.

8.11.4 Conversion from Distributed Data Parallel (DDP) application

Distributed Data Parallel (DDP) in torch.distributed module is a wrapper to help convert a single-worker training to distributed training. To convert from torch.distributed Distributed Data Parallel (DDP) application to PyTorch Neuron, first convert the application back to single-worker training, which simply involves removing the DDP wrapper, for example `model = DDP(model, device_ids=[rank])`. After this, follow the previous section to change to multi-worker training.

8.11.5 PyTorch Neuron environment variables

Environment variables allow modifications to PyTorch Neuron behavior without requiring code change to user script. See *PyTorch Neuron environment variables* for more details.

8.11.6 Persistent cache for compiled graphs

PyTorch/XLA has an internal in-memory compilation cache that caches previously compiled graphs within the same python process. However this internal cache is not persistent between runs. PyTorch Neuron includes a persistent cache that enables caching of previously compiled graph on disk so that subsequent run of the same program do not incur long compilation time. This cache is enabled by default and the default cache directory is /var/tmp/neuron-compile-cache.

The cache uses hash of the Neuron compiler flags and XLA graph as the key. If the Neuron compiler version or XLA graph changes, you will see recompilation. Examples of changes that would cause XLA graph change include:

- Model type and size
- Batch size
- Optimizer and optimizer hyperparameters
- Location of xm.mark_step()

To disable the cache, you can pass `--no_cache` option via NEURON_CC_FLAGS:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --no_cache'
```

To change the cache's root directory, pass `cache_dir=<root dir>` option via NEURON_CC_FLAGS (the actual cache directory will be in `<root dir>/neuron-compile-cache`:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_dir=
↪<root dir>'
```

Stale cached compiled graphs (NEFFs) are deleted from the cache whenever the size of cache is above default cache size of 100GB . The deletion order is based on least-recently-used first. To change the cache size, pass `--cache_size=SIZE_IN_BYTES`. For example, to change the cache size to 16 MB:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_
↪size=16777216'
```

A cache entry considered stale if the last used time is older than a time-to-live value, currently default to 30 days. If the last used time is earlier than the time-to-live value, then it is not deleted even if cache size exceeds cache size limit. To change cache time-to-live, set the option `--cache_ttl` to the number of days desired:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_ttl=60'
```

If in some cases, the compilation failed because of an environment issue, and you want to retry compilation, you can do so by adding `--retry_failed_compilation`. This will retry the compilation even if there is a failed NEFF in the cache.

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --retry_failed_
↪compilation'
```

You can change the verbose level of the compiler by adding `log_level` to either `WARNING`, `INFO` or `ERROR`. This can be done as follows:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --log_
↪level=INFO'
```

8.11.7 Number of graphs

PyTorch/XLA converts PyTorch's eager mode execution to lazy-mode graph-based execution. During this process, there can be multiple graphs compiled and executed if there are extra mark-steps or functions with implicit mark-steps. Additionally, more graphs can be generated if there are different execution paths taken due to control-flows.

8.11.8 Automatic casting of float tensors to BFloat16

With PyTorch Neuron, the default behavior is for torch.float (FP32) and torch.double (FP64) tensors to be mapped to torch.float in hardware. To reduce memory footprint and improve performance, torch.float and torch.double tensors can automatically be converted to BFloat16 by setting the environment variable `XLA_USE_BF16=1`. Alternatively, torch.float can automatically be converted to BFloat16 and torch.double converted to FP32 by setting the environment variable `XLA_DOWNCAST_BF16=1`.

8.11.9 Automatic Mixed-Precision

BF16 mixed-precision using PyTorch Autocast

By default, the compiler automatically cast internal FP32 operations to BF16. You can disable this and allow PyTorch's BF16 mixed-precision to do the casting. PyTorch's BF16 mixed-precision is achieved by casting certain operations to operate BF16. We currently use CUDA's list of operations that can operate in BF16:

(NOTE: Although convolution is in the list below, it is currently unsupported by Neuron. See model-architecture-fit)

```
_convolution
_convolution
_convolution_nogroup
conv1d
conv2d
conv3d
conv_tbc
conv_transpose1d
```

(continues on next page)

(continued from previous page)

```
conv_transpose2d
conv_transpose3d
convolution
cudnn_convolution
cudnn_convolution_transpose
cudnn_convolution
cudnn_convolution_transpose
cudnn_convolution
cudnn_convolution_transpose
prelu
addmm
addmv
addr
matmul
mm
mv
linear
addbmm
baddbmm
bmm
chain_matmul
linalg_multi_dot
```

To enable PyTorch's BF16 mixed-precision, first turn off the Neuron compiler auto-cast:

```
os.environ["NEURON_CC_FLAGS"] = "--auto-cast=none"
```

Next, overwrite torch.cuda.is_bf16_supported to return True:

```
torch.cuda.is_bf16_supported = lambda: True
```

Next, per recommendation from official PyTorch documentation, place only the forward-pass of the training step in the torch.autocast scope:

```
with torch.autocast(dtype=torch.bfloat16, device_type='cuda'):
    # forward pass
```

The device type is CUDA because we are using CUDA's list of BF16 compatible operations as mentioned above.

Example showing the original training code snippet:

```
def train_loop_fn(train_loader):
    for i, data in enumerate(train_loader):
        inputs = data[0]
        labels = data[3]
        outputs = model(inputs, labels=labels)
        loss = outputs.loss/ flags.grad_acc_steps
        loss.backward()
        optimizer.step()
        xm.mark_step()
```

The following shows the training loop modified to use BF16 autocast:

```
os.environ["NEURON_CC_FLAGS"] = "--auto-cast=none"

def train_loop_fn(train_loader):
    for i, data in enumerate(train_loader):
        torch.cuda.is_bf16_supported = lambda: True
        with torch.autocast(dtype=torch.bfloat16, device_type='cuda'):
            inputs = data[0]
            labels = data[3]
            outputs = model(inputs, labels=labels)
        loss = outputs.loss/ flags.grad_acc_steps
        loss.backward()
        optimizer.step()
        xm.mark_step()
```

For a full example of BF16 mixed-precision, see *PyTorch Neuron BERT Pretraining Tutorial*.

See official PyTorch documentation for more details about torch.autocast .

8.11.10 Tips and Best Practices

Understand the lazy mode in PyTorch Neuron

One significant difference between PyTorch Neuron and native PyTorch is that the PyTorch Neuron system runs in lazy mode while the native PyTorch runs in eager mode. Tensors in lazy mode are placeholders for building the computational graph until they are materialized after the compilation and evaluation are complete. The PyTorch Neuron system builds the computational graph on the fly when you call PyTorch APIs to build the computation using tensors and operators. The computational graph gets compiled and executed when `xm.mark_step()` is called explicitly or implicitly by `pl.MpDeviceLoader/pl.ParallelLoader`, or when you explicitly request the value of a tensor such as by calling `loss.item()` or `print(loss)`.

Minimize the number of compilation-and-executions

For best performance, you should keep in mind the possible ways to initiate compilation-and-executions as described in *Understand the lazy mode in PyTorch/XLA* and should try to minimize the number of compilation-and-executions. Ideally, only one compilation-and-execution is necessary per training iteration and is initiated automatically by `pl.MpDeviceLoader/pl.ParallelLoader`. The `MpDeviceLoader` is optimized for XLA and should always be used if possible for best performance. During training, you might want to examine some intermediate results such as loss values. In such case, the printing of lazy tensors should be wrapped using `xm.add_step_closure()` to avoid unnecessary compilation-and-executions.

Ensure common initial weights across workers

To achieve best accuracy during data parallel training, all workers need to have the same initial parameter states. This can be achieved by using the same seed across the workers. In the case of HuggingFace library, the set_seed function can be used. (https://github.com/pytorch/xla/issues/3216).

Use PyTorch/XLA's model save function

To avoid problems with saving and loading checkpoints, make sure you use PyTorch/XLA's model save function to properly checkpoint your model. For more information about the function, see torch_xla.core.xla_model.save in the *PyTorch on XLA Devices* documentation.

When training using multiple devices, `xla_model.save` can result in high host memory usage. If you see such high usage causing the host to run out of memory, please use torch_xla.utils.serialization.save . This would save the model in a serialized manner. When saved using the `serialization.save` api, the model should be loaded using `serialization.load` api. More information on this here: Saving and Loading Tensors

8.11.11 FAQ

What is the difference between Trainium and Inferentia?

Trainium is an accelerator designed to speed up training, whereas Inferentia is an accelerator designed to speed up inference.

8.11.12 Debugging and troubleshooting

To debug on PyTorch Neuron, please follow the debug guide.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.12 How to debug models in PyTorch Neuron (`torch-neuronx`)

Table of Contents

- *Printing Metrics*
- *Printing Tensors*
- *Use mark_step*
- *Using Eager Debug Mode*
- *Profiling Model Run*
- *Snapshotting*
  - *Snapshot FAQs:*

Torch-XLA evaluates operations lazily, which means it builds a symbolic graph in the background and the graph is executed in hardware only when the users request (print) for the output or a mark_step is encountered. To effectively debug training scripts with torch-xla, please use one of the approaches mentioned below:

8.12.1 Printing Metrics

Torch-xla provides a utility that records metrics of different sections of the code. These metrics can help figure out things like: How much time is spent in compilation? How much time is spent in execution? To check the metrics:

1. Import metrics: `import torch_xla.debug.metrics as met`
2. Print metrics at the end of the step: `print(met.metrics_report())`

Printing metrics should produce an output that looks like this:

```
Metric: CompileTime
  TotalSamples: 1
  Accumulator: 09s969ms486.408us
  Percentiles: 1%=09s969ms486.408us; 5%=09s969ms486.408us; 10%=09s969ms486.408us; 20
↪%=09s969ms486.408us; 50%=09s969ms486.408us; 80%=09s969ms486.408us; 90%=09s969ms486.
↪408us; 95%=09s969ms486.408us; 99%=09s969ms486.408us
.....
Metric: ExecuteTime
  TotalSamples: 1
  Accumulator: 186ms062.970us
  Percentiles: 1%=186ms062.970us; 5%=186ms062.970us; 10%=186ms062.970us; 20%=186ms062.
↪970us; 50%=186ms062.970us; 80%=186ms062.970us; 90%=186ms062.970us; 95%=186ms062.970us;␣
↪99%=186ms062.970us
....
Metric: TensorsGraphSize
  TotalSamples: 1
  Accumulator: 9.00
  Percentiles: 1%=9.00; 5%=9.00; 10%=9.00; 20%=9.00; 50%=9.00; 80%=9.00; 90%=9.00; 95%=9.
↪00; 99%=9.00
Metric: TransferFromServerTime
  TotalSamples: 2
  Accumulator: 010ms130.597us
  ValueRate: 549ms937.108us / second
  Rate: 108.372 / second
  Percentiles: 1%=004ms948.602us; 5%=004ms948.602us; 10%=004ms948.602us; 20%=004ms948.
↪602us; 50%=006ms181.995us; 80%=006ms181.995us; 90%=006ms181.995us; 95%=006ms181.995us;␣
↪99%=006ms181.995us
Metric: TransferToServerTime
  TotalSamples: 6
  Accumulator: 061ms698.791us
  ValueRate: 007ms731.182us / second
  Rate: 0.665369 / second
  Percentiles: 1%=006ms848.579us; 5%=006ms848.579us; 10%=006ms848.579us; 20%=007ms129.
↪666us; 50%=008ms940.718us; 80%=008ms496.166us; 90%=024ms636.413us; 95%=024ms636.413us;␣
↪99%=024ms636.413us
Metric: TransferToServerTransformTime
  TotalSamples: 6
  Accumulator: 011ms835.717us
  ValueRate: 001ms200.844us / second
  Rate: 0.664936 / second
  Percentiles: 1%=108.403us; 5%=108.403us; 10%=108.403us; 20%=115.676us; 50%=167.399us;␣
↪80%=516.659us; 90%=010ms790.400us; 95%=010ms790.400us; 99%=010ms790.400us
.....
Counter: xla::_copy_from
```

(continues on next page)

(continued from previous page)

```
  Value: 7
Counter: xla::addmm
  Value: 2
Counter: xla::empty
  Value: 5
Counter: xla::t
  Value: 2
....
Metric: XrtCompile
  TotalSamples: 1
  Accumulator: 09s946ms607.609us
  Mean: 09s946ms607.609us
  StdDev: 000.000us
  Percentiles: 25%=09s946ms607.609us; 50%=09s946ms607.609us; 80%=09s946ms607.609us; 90
↪%=09s946ms607.609us; 95%=09s946ms607.609us; 99%=09s946ms607.609us
Metric: XrtExecute
  TotalSamples: 1
  Accumulator: 176ms932.067us
  Mean: 176ms932.067us
  StdDev: 000.000us
  Percentiles: 25%=176ms932.067us; 50%=176ms932.067us; 80%=176ms932.067us; 90%=176ms932.
↪067us; 95%=176ms932.067us; 99%=176ms932.067us
Metric: XrtReadLiteral
  TotalSamples: 2
  Accumulator: 608.578us
  Mean: 304.289us
  StdDev: 067.464us
  Rate: 106.899 / second
  Percentiles: 25%=236.825us; 50%=371.753us; 80%=371.753us; 90%=371.753us; 95%=371.753us;
↪ 99%=371.753us
```

As seen, you can get useful information about graph compile times/execution times. You can also know which operators are present in the graph, which operators are run on the CPU and which operators are run on an XLA device. For example, operators that have a prefix `aten::` would run on the CPU, since they do not have xla lowering. All operators with prefix `xla::` would run on an XLA device. Note: aten operators that do not have xla lowering would result in a graph fragmentation and might end up slowing down the entire execution. If you encounter such operators, create a request for operator support.

8.12.2 Printing Tensors

Users can print tensors in their script as below:

```
import os
import torch
import torch_xla
import torch_xla.core.xla_model as xm

device = xm.xla_device()
input1 = torch.randn(2,10).to(device)
# Defining 2 linear layers
linear1 = torch.nn.Linear(10,30).to(device)
```

(continues on next page)

(continued from previous page)

```
linear2 = torch.nn.Linear(30,20).to(device)

# Running forward
output1 = linear1(input1)
output2 = linear2(output1)
print(output2)
```

Since torch-xla evaluates operations lazily, when you try to print `output2` , the graph associated with the tensor would be evaluated. When a graph is evaluated, it is first compiled for the device and executed on the selected device. Note: Each tensor would have a graph associated with it and can result in graph compilations and executions. For example, in the above script, if you try to print `output1` , a new graph is cut and you would see another evaluation. To avoid multiple evaluations, you can make use of `mark_step` (next section).

8.12.3 Use mark_step

Torch-XLA provides an api called `mark_step` which evaluates a graph collected upto that point. While this is similar to printing of an output tensor wherein a graph is also evaluated, there is a difference. When an output tensor is printed, only the graph associated with that specific tensor is evaluated, whereas mark_step enables all the output tensors up to `mark_step` call to be evaluated in a single graph. Hence, any tensor print after `mark_step` would be effectively free of cost as the tensor values are already evaluated. Consider the example below:

```
import os
import torch
import torch_xla
import torch_xla.core.xla_model as xm
import torch_xla.debug.metrics as met

device = xm.xla_device()
input1 = torch.randn(2,10).to(device)
# Defining 2 linear layers
linear1 = torch.nn.Linear(10,30).to(device)
linear2 = torch.nn.Linear(30,20).to(device)

# Running forward
output1 = linear1(input1)
output2 = linear2(output1)
xm.mark_step()
print(output2)
print(output1)
# Printing the metrics to check if compilation and execution occurred
print(met.metrics_report())
```

In the printed metrics, the number of compiles and executions is only 1, even though 2 tensors are printed. Hence, to avoid multiple graph evaluations, it is recommended that you visualize tensors after a `mark_step` . You can also make use of the add_step_closure api for this purpose. With this api, you pass in the tensors that needs to be visualized/printed. The added tensors would then be preserved in the graph and can be printed as part of the callback function passed to the api. Here is a sample usage: https://github.com/pytorch/xla/blob/master/test/test_train_mp_mnist.py#L133

Note: Graph compilations can take time as the compiler optimizes the graph to run on device.

8.12.4 Using Eager Debug Mode

Eager debug mode provides a convenient utility to step through the code and evaluate operators one by one for correctness. Eager debug mode is useful to inspect your models the way you would do in eager-mode frameworks like PyTorch and Tensorflow. With Eager Debug Mode operations are executed eagerly. As soon as an operation is registered with torch-xla, it would be sent for compilation and execution. Since compiling a single operation, the time spent would be minimal. Moreover, the chances of hitting the framework compilation cache increases as models would have repeated operations throughout. Consider example 1 below:

```
# Example 1

import os
# You need to set this env variable before importing torch-xla
# to run in eager debug mode.
os.environ["NEURON_USE_EAGER_DEBUG_MODE"] = "1"

import torch
import torch_xla
import torch_xla.core.xla_model as xm
import torch_xla.debug.metrics as met

device = xm.xla_device()
input1 = torch.randn(2,10).to(device)
# Defining 2 linear layers
linear1 = torch.nn.Linear(10,30).to(device)
linear2 = torch.nn.Linear(30,20).to(device)

# Running forward
output1 = linear1(input1)
output2 = linear2(output1)

# Printing the metrics to check if compilation and execution occurred
# Here, in the metrics you should notice that the XRTCompile and XRTExecute
# value is non-zero, even though no tensor is printed. This is because, each
# operation is executed eagerly.
print(met.metrics_report())

print(output2)
print(output1)
# Printing the metrics to check if compilation and execution occurred.
# Here the XRTCompile count should be same as the previous count.
# In other words, printing tensors did not incur any extra compile
# and execution of the graph
print(met.metrics_report())
```

As seen from the above scripts, each operator is evaluated eagerly and there is no extra compilation when output tensors are printed. Moreover, together with the on-disk Neuron persistent cache, eager debug mode only incurs one time compilation cost when the ops is first run. When the script is run again, the compiled ops will be pulled from the persistent cache. Any changes you make to the training script would result in the re-compilation of only the newly inserted operations. This is because each operation is compiled independently. Consider example 2 below:

```
# Example 2

import os
```

(continues on next page)

(continued from previous page)

```
# You need to set this env variable before importing torch-xla
# to run in eager debug mode.
os.environ["NEURON_USE_EAGER_DEBUG_MODE"] = "1"

import torch
import torch_xla
import torch_xla.core.xla_model as xm
import torch_xla.debug.metrics as met

os.environ['NEURON_CC_FLAGS'] = "--log_level=INFO"

device = xm.xla_device()
input1 = torch.randn(2,10).to(device)
# Defining 2 linear layers
linear1 = torch.nn.Linear(10,30).to(device)
linear2 = torch.nn.Linear(30,20).to(device)
linear3 = torch.nn.Linear(20,30).to(device)
linear4 = torch.nn.Linear(30,20).to(device)

# Running forward
output1 = linear1(input1)
output2 = linear2(output1)
output3 = linear3(output2)

# Note the number of compiles at this point and compare
# with the compiles in the next metrics print
print(met.metrics_report())

output4 = linear4(output3)
print(met.metrics_report())
```

Running the above example 2 script after running example 1 script, you may notice that from the start until the statement `output2 = linear2(output1)` , all the graphs would hit the persistent cache. Executing the line `output3 = linear3(output2)` would result in a new compilation for `linear3` layer only because the layer configuration is new. Now, when we run `output4 = linear4(output3)` , you would observe no new compilation happens. This is because the graph for `linear4` is same as the graph for `linear2` and hence the compiled graph for `linear2` is reused for `linear4` by the framework's internal cache.

Eager debug mode avoids the wait times involved with tensor printing because of larger graph compilation. It is designed only for debugging purposes, so when the training script is ready, please remove the `NEURON_USE_EAGER_DEBUG_MODE` environment variable from the script in order to obtain optimal performance.

By default, in eager debug mode the logging level in the Neuron compiler is set to error mode. Hence, no logs would be generated unless there is an error. Before your first print, if there are many operations that needs to be compiled, there might be a small delay. In case you want to check the logs, switch on the `INFO` logs for compiler using:

```
os.environ["NEURON_CC_FLAGS"] = "--log_level=INFO"
```

8.12.5 Profiling Model Run

Profiling model run can help to identify different bottlenecks and resolve issues faster. You can profile different sections of the code to see which block is the slowest. To profile model run, you can follow the steps below:

1. Add: `import torch_xla.debug.profiler as xp`
2. Start server. This can be done by adding the following line after creating xla device: `server = xp.start_server(9012)`
3. In a separate terminal, start tensorboard. The logdir should be in the same directory from which you run the script.

   Open the tensorboard on a browser. Go to profile section in the top right. Note: you may have to install the profile plugin using: `pip install tensorboard-plugin-profile`

4. When you click on the profile, it should give an option to capture profile. Clicking on capture profile produces the following pop-up.

In the URL enter: `localhost:9012` . Port in this URL should be same as the one you gave when starting the server in the script.

5. Once done, click capture and it should automatically load the following page:

6. To check the profile for different blocks of code, head to `trace_viewer` under `Tools` (on the left column).

7. It should show a profile that looks like this:

Note: By default, torch-xla would time different blocks of code inside the library. However, you can also profile block of code in your scripts. This can be done by adding the code within a `xp.Trace` context as follows:

```
....
for epoch in range(total_epochs):
    inputs = torch.randn(1,10).to(device)
    labels = torch.tensor([1]).to(device)
    with xp.Trace("model_build"):
        loss = model(inputs, labels)
```

(continues on next page)

(continued from previous page)

```
    with xp.Trace("loss_backward"):
        loss.backward()
....
```

It should produce a profile that has the `model_build` and `loss_backward` section timed. This way you can time any block of script for debugging.

Note: If you are running your training script in a docker container, to view the tensorboard, you should launch the docker container using flag: `--network host` eg. `docker run --network host my_image:my_tag`

8.12.6 Snapshotting

Snapshotting models can be used to dump debug information that can then be sent to the Neuron team. Neuron execution relies on a series of compiled graphs. Internally, graph HLOs are used as an intermediate representation which is then compiled. Then, during execution, the graph inputs are passed to the Neuron runtime, which produces outputs using the compiled graph. Snapshotting saves the inputs to a graph execution, executes the graphs, saves the outputs of the execution, and then bundles and dumps the inputs, outputs and graph HLO in one file. This is illustrated here:

This feature can be enabled using the following environment variables, which can be set at the beginning of your script as follows.

```
....
os.environ["XLA_FLAGS"] = " --xla_dump_to=dump"
os.environ["NEURON_FRAMEWORK_DEBUG"] = "1"
os.environ["NEURON_DUMP_HLO_SNAPSHOT"] = "1"
....
```

This set of environment variables will produce snapshots under the dump folder with the extensions `.hlo.snapshot.pb` or `.decomposed_hlo_snapshot` at every iteration. For example a file that looks like the following would be produced.

```
dump/module_SyncTensorsGraph.387.pid_12643.execution_7496.hlo_snapshot.pb
```

The dumping environment variable can be set and unset at specific iterations as shown in the following example.

```
....
for step in range(STEPS):
    if step == 20:
        os.environ["NEURON_DUMP_HLO_SNAPSHOT"] = "1"
    else:
        os.environ.pop('NEURON_DUMP_HLO_SNAPSHOT', None)
    train_x = torch.randn(BATCH_SIZE, 28, 28)
    train_x = train_x.to(device)
    loss = model(train_x)
    loss.backward()
    optimizer.step()
    xm.mark_step()
....
```

Additionally, we provide capabilities to snapshot graphs automatically. The environment variables above can be set as follows:

```
....
os.environ["XLA_FLAGS"] = " --xla_dump_to=dump"
os.environ["NEURON_FRAMEWORK_DEBUG"] = "1"
os.environ["NEURON_DUMP_HLO_SNAPSHOT"] = "ON_NRT_ERROR"
....
```

When unexpected errors such as a graph execution producing NaNs occurs, snapshots will be automatically produced and execution will be terminated. Occasionally, for larger models, automatic snapshotting may not capture snapshots due to the device memory being exhausted. In this case, the above flag can be set to `os.environ["NEURON_DUMP_HLO_SNAPSHOT"] = "ON_NRT_ERROR_HYBRID"`, this will allocate memory for inputs on both the device and host memory. In some additional cases, this may still go out of memory and may need to be set to `os.environ["NEURON_DUMP_HLO_SNAPSHOT"] = "ON_NRT_ERROR_CPU"` to avoid allocating any memory on the device at all for automatic snapshotting.

Snapshot FAQs:

When should I use this features?

This feature should be used when debugging errors that requires interfacing with and providing debug data to the Neuron team. Snapshotting may be redundant and unnecessary in some situations. For example, when only the model weights are necessary for debugging, methods such as checkpointing may be more convenient to use.

What sort of data is captured with these snapshots?

The type of data captured by these snapshots may include model graphs in HLO form, weights/parameters, optimizer states, intermediate tensors and gradients. This data may be considered sensitive and this should be taken into account before sending the data to the Neuron team.

What is the size of these snapshots?

The size of snapshots can be significant for larger models such as GPT or BERT with several GBs worth of data for larger graphs, so it is recommended to check that sufficient disk space exists before using snapshotting. In addition, limiting the amount of snapshots taken in a run will help to preserve disk space.

Will snapshotting add overhead to my execution?

Snapshotting does add a small overhead to the execution in most cases. This overhead can be significant if snapshots are dumped at every iteration. In order to alleviate some of this overhead, in the case that snapshotting is not necessary on all cores the following environment variable can be set to collect snapshots only on the first core. In addition, checkpointing in tandem with snapshotting can be useful to reduce overhead. A checkpoint close to the problem iteration can be captured, then execution resumed with snapshots enabled.

```
....
os.environ["NEURON_NC0_ONLY_SNAPSHOT"] = "1"
....
```

How can I share snapshots with the Neuron team?

These snapshots can be shared with the Neuron team via S3 bucket.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.13 Developer Guide for Profiling with PyTorch Neuron (torch-neuronx)

Table of Contents

- *Introduction*
- *Example used in this guide*
  - *Prerequisites*
  - *Environment*
  - *Setup*
- *Viewing the Trace on TensorBoard*
- *Using Named Blocks for the Trace*

8.13.1 Introduction

The Neuron PyTorch profiler is a context manager wrapping around the entire model and training loop. Specifically this is the context manager: `torch_neuronx.experimental.profiler.profile`. This is a wrapper of the XLA Debug Profiler which we imported earlier as `import torch_xla.debug.profiler as xp`, and is backwards-compatible. Here are the parameters of the profiler context manager:

1. `port`: Port to run the profiling GRPC server on. Default is 9012.
2. `profile_type`: There is "trace" and "operator". "trace" is the Torch Runtime Trace Level, while "operator" is the Model Operator Trace Level.
3. `ms_duration`: This defines how long the profiler will capture the HLO artifacts from the model to view in the profiler. The unit is in milliseconds.
4. `neuron_tensorboard_plugin_dir`: The directory the neuron tensorboard plugin will file write to (NB: Assumes that the tensorboard logdir="log/")
5. `delete_working`: If set to False turns off the deletion of temporary files (default True)

We move the model to the xla device *inside the context manager.* This is important, as this allows the profiler to collect the operations and processes from the `neuronx-cc` compiler artifacts. If the model is moved to the xla device outside of the context manager, the profiling won't work.

Note: The warnings about the `XLA_IR_DEBUG` and `XLA_HLO_DEBUG` env vars not being set can be ignored for the most part. This warning only comes into play when compiling the model for Neuron outside of the profiler context manager.

After running this script, notice a `./logs` directory has been created. It contains the TensorBoard logs including the profiler views.

8.13.2 Example used in this guide

We will use the following code sample to describe in detail how to use the Neuron PyTorch profiling API.

Prerequisites

1. Initial Trn1 setup for PyTorch (torch-neuronx) has been done

Environment

```
#activate python virtual environment and install tensorboard_plugin_neuron
source ~/aws_neuron_venv_pytorch_p38/bin/activate
pip install tensorboard_plugin_neuronx

#create work directory for the Neuron Profiling tutorials
mkdir -p ~/neuron_profiling_tensorboard_examples
cd ~/neuron_profiling_tensorboard_examples
```

Setup

Create a new working directory:

```
mkdir simple_demo
cd simple_demo
```

Save the following code as `demo.py`:

```
import os

import torch
import torch.nn as nn
import torch.nn.functional as F

# XLA imports
import torch_xla
import torch_xla.core.xla_model as xm
import torch_xla.debug.profiler as xp

import torch_neuronx
from torch_neuronx.experimental import profiler

os.environ["NEURON_CC_FLAGS"] = "--cache_dir=./compiler_cache"

# Global constants
EPOCHS = 10

# Declare 3-layer MLP Model
class MLP(nn.Module):
  def __init__(self, input_size = 10, output_size = 2, layers = [5, 5]):
      super(MLP, self).__init__()
      self.fc1 = nn.Linear(input_size, layers[0])
```

(continues on next page)

(continued from previous page)

```
        self.fc2 = nn.Linear(layers[0], layers[1])
        self.fc3 = nn.Linear(layers[1], output_size)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return F.log_softmax(x, dim=1)


def main():
    # Fix the random number generator seeds for reproducibility
    torch.manual_seed(0)

    # XLA: Specify XLA device (defaults to a NeuronCore on Trn1 instance)
    device = xm.xla_device()

    # Start the proflier context-manager
    with torch_neuronx.experimental.profiler.profile(
        port=9012,
        profile_type='trace',
        ms_duration=15000 ) as profiler:

        # IMPORTANT: the model has to be transferred to XLA within
        # the context manager, otherwise profiling won't work
        model = MLP().to(device)
        optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
        loss_fn = torch.nn.NLLLoss()

        # start training loop
        print('----------Training ---------------')
        model.train()
        for epoch in range(EPOCHS):
            optimizer.zero_grad()
            train_x = torch.randn(1,10).to(device)
            train_label = torch.tensor([1]).to(device)

            #forward
            loss = loss_fn(model(train_x), train_label)

            #back
            loss.backward()
            optimizer.step()

            # XLA: collect ops and run them in XLA runtime
            xm.mark_step()

    print('----------End Training ---------------')

if __name__ == '__main__':
    main()
```

Then run it!

```
python demo.py
```

8.13.3 Viewing the Trace on TensorBoard

To view the TensorBoard logs, run `tensorboard --logdir=./logs`

Note: Depending on TensorBoard version `--load_fast=false` might be an additional parameter to add to view the trace.

Take note of the port (usually 6006) and enter `localhost:<port>` into the local browser (assuming port forwarding is set up properly):

```
TensorBoard 1.15.0 at http://ip-172-31-33-242:6006/ (Press CTRL+C to quit)
```

Once `localhost:<port>` is entered, verify that the "NEURON" view is shown:

If "NEURON" isn't shown on the top left hand side, select "NEURON" from the drop down on the top right hand side

On the Left Hand Side, there are two dropdown menus: Run & Tool.

The Run dropdown would contain the Torch Runtime Trace and Operator Level Trace views; however since we only ran the "trace" (i.e Torch Runtime Trace Level), we'll only see that log. The Torch Runtime Trace views are simply dates in `year_month_day_hour_minute_second_millisecond` format. The Tool Dropdown only contains the "trace" option.

The trace view should look like this:

Let's zoom into the following section of the trace:

After zooming in the trace should look like this:

Notice on the top, there is a `StepMarker` process followed by `NeuronDevice Execution` process. This correlates to the `xm.mark_step()` call which executes the collected graph of our model on Neuron. For the Operator Level Trace ("operator"), we'll be profiling the model operators that occur on Neuron. In other words, the profiler will zoom into the `NeuronDevice Execution` process, if the user specifies `profile_type='trace'`.

8.13.4 Using Named Blocks for the Trace

What we've produced so far is the default behavior of the profiler, however it would be more useful to profile specific blocks of our code to narrow down onto performance bottlenecks. To do this, use `xp.Trace` context manager. Replace the respective code in the training loop with the following:

```
...
optimizer.zero_grad()
train_x = torch.randn(1,10).to(device)
train_label = torch.tensor([1]).to(device)

with xp.Trace("model_build"):
    loss = loss_fn(model(train_x), train_label)
with xp.Trace("loss_backward"):
    loss.backward()
with xp.Trace("optimizer_step"):
    optimizer.step()

# XLA: collect ops and run them in XLA runtime
xm.mark_step()
...
```

Run the script, and follow the same TensorBoard steps. Afterwards, the trace should look like this:

As seen, the `model_build`, `loss_backward` and `optimizer_step` sections have been profiled.

Note: If you are running your training script in a docker container, to view the tensorboard, you should launch the

docker container using flag: `-network host` eg. `docker run -network host my_image:my_tag`

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.14 PyTorch Neuron (`torch-neuronx`) - Training Supported Operators

Table of Contents

- *Operator support*

8.14.1 Operator support

The following list the aten operators supported by torch-neuronx.

aten::_s_where
aten::_softmax
aten::_softmax_backward_data
aten::_unsafe_view
aten::add
aten::addcdiv_
aten::addcmul
aten::addmm
aten::bernoulli_
aten::bmm
aten::constant_pad_nd
aten::div
aten::embedding
aten::embedding_dense_backward
aten::empty
aten::expand
aten::fill_
aten::index_select
aten::_log_softmax
aten::_log_softmax_backward_data
aten::lt
aten::mm
aten::mul
aten::native_batch_norm
aten::native_batch_norm_backward
aten::neg
aten::permute
aten::relu
aten::rsub
aten::select
aten::slice

continues on next page

Table 1 – continued from previous page

aten::sqrt
aten::sum
aten::t
aten::tanh
aten::tanh_backward
aten::threshold_backward
aten::transpose
aten::unsqueeze
aten::view
aten::zero_

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.15 PyTorch Neuron (`torch-neuronx`) for Training Troubleshooting Guide

Table of contents

- *General Troubleshooting*
- *Possible Error Conditions*
  - *Non-Fatal Error OpKernel ('op: "TPU*" device_type: "CPU"')*
  - *XLA runtime error: "Invalid argument: Cannot assign a device for operation"*
  - *Error: "Could not start gRPC server"*
  - *Failed compilation result in the cache*
  - *Compilation error: "Expect ap datatype to be of type float32 float16 bfloat16 uint8"*
  - *NeuronCore(s) not available - Requested:1 Available:0*
  - *TDRV error "TDRV:exec_consume_infer_status_notification"*
  - *TDRV error "TDRV:tdrv_one_tmpbuf_reserve Number of ONE TMPBUF pages requested exceeded the max number of pages allowed (requested: <N>, max allowed: 16)."*
  - *Could not open the ndX, close device failed, TDRV not initialized*
  - *Multiworker execution hangs during NCCL init*
  - *NRT init error "One or more engines are running. Please restart device by reloading driver"*
  - *NRT error "ERROR TDRV:kbl_model_add Attempting to load an incompatible model!"*
  - *NRT error "ERROR HAL:aws_hal_sprot_config_remap_entry SPROT remap destination address must be aligned size"*
  - *NCCL warning : "NCCL WARN Timeout waiting for RX (waited 120 sec) - retrying"*
  - *RPC error: "RPC failed with status = 'UNAVAILABLE: Socket closed'"*
  - *Error "Assertion `listp->slotinfo[cnt].gen <= GL(dl_tls_generation)' failed" followed by 'RPC failed with status = "UNAVAILABLE: Connection reset by peer"'*

- *RPC connection error: "RPC failed with status = UNAVAILABLE: Connection reset by peer" not preceded by any error*
- *Runtime errors "Missing infer_status notification" followed by "inference timeout"*
- *Protobuf Error "TypeError: Descriptors cannot not be created directly."*
- *TDRV error "Timestamp program stop timeout"*
- *Compiler error "module 'numpy' has no attribute 'asscalar'"*
- *Import errors 'generic_type: type "IrValue" is already registered!' or 'generic_type: type "XlaBuilder" is already registered!'*
- *Import error "import _XLAC ImportError: <>/site-packages/_XLAC.cpython-38-x86_64-linux-gnu.so: undefined symbol"*

This document shows common issues users may encounter while using PyTorch-Neuron and provides guidance how to resolve or work-around them.

8.15.1 General Troubleshooting

For XLA-related troubleshooting notes see *How to debug models in PyTorch Neuron* and PyTorch-XLA troubleshooting guide.

If your multi-worker training run is interrupted, you may need to kill all the python processes (WARNING: this kills all python processes and reload the driver):

```
killall -9 python
killall -9 python3
sudo rmmod neuron; sudo modprobe neuron
```

To turn on RT debug:

```
os.environ["NEURON_RT_LOG_LEVEL"] = "INFO"
```

To turn on Neuron NCCL debug:

```
os.environ["NCCL_DEBUG"] = "WARN"
os.environ["NCCL_DEBUG_SUBSYS"] = "ALL"
```

8.15.2 Possible Error Conditions

Non-Fatal Error OpKernel ('op: "TPU*" device_type: "CPU"')

During execution using PyTorch Neuron, you may see these non-fatal error messages:

```
E tensorflow/core/framework/op_kernel.cc:1676] OpKernel ('op: "TPURoundRobin" device_
↪type: "CPU"') for unknown op: TPURoundRobin
E tensorflow/core/framework/op_kernel.cc:1676] OpKernel ('op: "TpuHandleToProtoKey"␣
↪device_type: "CPU"') for unknown op: TpuHandleToProtoKey
```

They don't affect operation of the PyTorch Neuron and can be ignored.

XLA runtime error: "Invalid argument: Cannot assign a device for operation"

```
RuntimeError: tensorflow/compiler/xla/xla_client/xrt_computation_client.cc:490 : Check␣
↪failed: session->session()->Run(session_work->feed_inputs, session_work->outputs_
↪handles, &outputs) == ::tensorflow::Status::OK() (INVALID_ARGUMENT: Cannot assign a␣
↪device for operation XRTAllocateFromTensor: {{node XRTAllocateFromTensor}} was␣
↪explicitly assigned to /job:localservice/replica:0/task:0/device:TPU:0 but available␣
↪devices are [ /job:localservice/replica:0/task:0/device:CPU:0, /job:localservice/
↪replica:0/task:0/device:TPU_SYSTEM:0, /job:localservice/replica:0/task:0/device:XLA_
↪CPU:0 ]. Make sure the device specification refers to a valid device.
     [[XRTAllocateFromTensor]] vs. OK)
  *** Begin stack trace ***
     tensorflow::CurrentStackTrace()

     xla::util::MultiWait::Complete(std::function<void ()> const&)

     clone
  *** End stack trace ***
```

The above error indicates that the framework was not able to initialize the neuron runtime. If you get the above error, check for the following:

1. No other process is taking the neuron cores. If yes, you may have to kill that process.
2. If no process is running, try reloading the driver using `sudo rmmod neuron; sudo modprobe neuron`

Error: "Could not start gRPC server"

If you get "Could not start gRPC server" error, please check if there are any leftover python processes from a previous interrupted run and terminate them before restarting run.

```
E0207 17:22:12.592127280   30834 server_chttp2.cc:40]        {"created":"@1644254532.
↪592081429","description":"No address added out of total 1 resolved","file":"external/
↪com_github_grpc_grpc/src/core/ext/t
ransport/chttp2/server/chttp2_server.cc","file_line":395,"referenced_errors":[{"created":
↪"@1644254532.592078907","description":"Failed to add any wildcard listeners","file":
↪"external/com_github_grpc_grpc/s
rc/core/lib/iomgr/tcp_server_posix.cc","file_line":342,"referenced_errors":[{"created":
↪"@1644254532.592072626","description":"Unable to configure socket","fd":10,"file":
↪"external/com_github_grpc_grpc/src/c
ore/lib/iomgr/tcp_server_utils_posix_common.cc","file_line":216,"referenced_errors":[{
↪"created":"@1644254532.592068939","description":"Address already in use","errno":98,
↪"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/tcp_server_utils_posix_common.
↪cc","file_line":189,"os_error":"Address already in use","syscall":"bind"}]},{"created":
↪"@1644254532.592078512","description":"Unable to configure socket"
,"fd":10,"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/tcp_server_utils_posix_
↪common.cc","file_line":216,"referenced_errors":[{"created":"@1644254532.592077123",
↪"description":"Address already in
 use","errno":98,"file":"external/com_github_grpc_grpc/src/core/lib/iomgr/tcp_server_
↪utils_posix_common.cc","file_line":189,"os_error":"Address already in use","syscall":
↪"bind"}]}]}]}
2022-02-07 17:22:12.592170: E tensorflow/core/distributed_runtime/rpc/grpc_server_lib.cc:
↪545] Unknown: Could not start gRPC server
```

Failed compilation result in the cache

All compilation results are by default saved in `Neuron Persistent Cache`. If the Neuron Compiler fails to compile a graph, we save the failed result in the cache. The reason for doing so is, if the user tries to run the same script, we want the users to error out early rather than wait for the compilation to progress and see an error at the later stage. However, there could be certain cases under which a failed compilation may be do you some environment issues. One possible reason of failure could be, during compilation the process went out of memory. This can happen if you are running multiple processes in parallel such that not enough memory is available for compilation of graph. Failure due to such reasons can be easily mitigated by re-running the compilation. In case, you want to retry a failed compilation, you can do that by passing `--retry_failed_compilation` as follows:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --retry_failed_
↪compilation'
```

This would retry the compilation and would replace a failed result in the cache with a successful compilation result.

Compilation error: "Expect ap datatype to be of type float32 float16 bfloat16 uint8"

If an XLA example fails to run because of failed compilation and one of the error messages is "Expect ap datatype to be of type float32 float16 bfloat16 uint8", then please set the environment variable `XLA_USE_32BIT_LONG=1` in your script:

```
os.environ['XLA_USE_32BIT_LONG'] = '1'
```

```
11/18/2021 04:51:25 PM WARNING 34567 [StaticProfiler]: matmul-based transposes inserted␣
↪by penguin takes up 93.66 percent of all matmul computation
terminate called after throwing an instance of 'std::runtime_error'
  what():  === BIR verification failed ===
Reason: Expect ap datatype to be of type float32 float16 bfloat16 uint8
Instruction: I-545-0
Opcode: Matmult
Input index: 0
Argument AP:
Access Pattern: [[1,8],[1,1],[1,1]]
Offset: 0
Memory Location: {compare.85-t604_i0}@SB<0,0>(8x2)#Internal DebugInfo: <compare.
↪85||uint16||UNDEF||[8, 1, 1]>
```

NeuronCore(s) not available - Requested:1 Available:0

When you see "NeuronCore(s) not available" please terminate processes that may be holding the NeuronCores and terminate any neuron-top sessions that are running. Also check if someone else is using the system. Then do "sudo rmmod neuron; sudo modprobe neuron" to reload the driver.

```
2021-Nov-15 15:21:28.0231 7245:7245 ERROR NRT:nrt_allocate_neuron_cores NeuronCore(s)␣
↪not available - Requested:nc1-nc1 Available:0
2021-11-15 15:21:28.231864: F ./tensorflow/compiler/xla/service/neuron/neuron_runtime.h:
↪1037] Check failed: status == NRT_SUCCESS NEURONPOC : nrt_init failed. Status = 1
```

Often when you run multi-worker training, there can be many python processes leftover after a run is interrupted. To kill all python processes, run the follow (WARNING: this kills all python processes on the system) then reload the driver:

```
killall -9 python
killall -9 python3
sudo rmmod neuron; sudo modprobe neuron
```

TDRV error "TDRV:exec_consume_infer_status_notification"

If you see TDRV error "TDRV:exec_consume_infer_status_notification", try reloading the driver using `sudo modprobe -r neuron; sudo modprobe neuron;`.

```
2022-Mar-10 18:51:19.07392022-Mar-10 18:51:19.0739 17821:17931 ERROR  TDRV:exec_consume_
↪infer_status_notifications  17822:18046 ERROR  TDRV:exec_consume_infer_status_
↪notifications Unexpected number of CC notifications:  mod->cc_op_count=1, cc_start_
↪cnt=0, cc_end_cnt=0Unexpected number of CC notifications:  mod->cc_op_count=1, cc_
↪start_cnt=0, cc_end_cnt=0

2022-Mar-10 18:51:19.07392022-Mar-10 18:51:19.0739 17821:17931 ERROR  TDRV:exec_consume_
↪infer_status_notifications  17822:18046 ERROR  TDRV:exec_consume_infer_status_
↪notifications (NON-FATAL, Ignoring) inference timeout (180000 ms) on Neuron Device 0␣
↪NC 0, waiting for cc status notifications.

(NON-FATAL, Ignoring) inference timeout (180000 ms) on Neuron Device 0 NC 1, waiting for␣
↪cc status notifications.
```

TDRV error "TDRV:tdrv_one_tmpbuf_reserve Number of ONE TMPBUF pages requested exceeded the max number of pages allowed (requested: <N>, max allowed: 16)."

If you see the TDRV error "TDRV:tdrv_one_tmpbuf_reserve Number of ONE TMPBUF pages requested exceeded the max number of pages allowed (requested: <N>, max allowed: 16)", it maybe due to model tensors requiring more device memory then available. A solution is to try training with a smaller data batch size.

```
ERROR  TDRV:tdrv_one_tmpbuf_reserve                 Number of ONE TMPBUF pages requested␣
↪exceeded the max number of pages allowed (requested: 28, max allowed: 16).
ERROR  TDRV:copy_and_stage_mr                       Failed to reserve one tmpbuf memory
ERROR  TDRV:kbl_model_add                           copy_and_stage_mr() error
W tensorflow/core/distributed_runtime/rpc/grpc_remote_master.cc:157] RPC failed with␣
↪status = "UNAVAILABLE: Socket closed" and grpc_error_string = "{"created":"@1669183391.
↪155135683","description":"Error received from peer ipv4:172.31.58.24:43941","file":
↪"external/com_github_grpc_grpc/src/core/lib/surface/call.cc","file_line":1056,"grpc_
↪message":"Socket closed","grpc_status":14}", maybe retrying the RPC
```

Could not open the ndX, close device failed, TDRV not initialized

If you see error messages stating "Could not open the ndX" (where X is an integer from 0..15), please run `neuron-ls` and ensure that you are able to see all 16 Neuron devices in the output. If one or more devices are missing please report the issue to aws-neuron-support@amazon.com with the instance ID and a screen capture of `neuron-ls` output.

```
2021-Nov-11 15:33:20.0161  7912:7912  ERROR  TDRV:tdrv_init_mla_phase1                 ␣
↪ Could not open the nd0
2021-Nov-11 15:33:20.0161  7912:7912  ERROR  TDRV:tdrv_destroy_one_mla                 ␣
↪ close device failed
```

(continues on next page)

(continued from previous page)

```
2021-Nov-11 15:33:20.0161  7912:7912  ERROR  TDRV:tdrv_destroy
↪ TDRV not initialized
2021-Nov-11 15:33:20.0161  7912:7912  ERROR   NRT:nrt_init
↪ Failed to initialize devices, error:1
2021-11-11 15:33:20.161331: F ./tensorflow/compiler/xla/service/neuron/neuron_runtime.h:
↪1033] Check failed: status == NRT_SUCCESS NEURONPOC : nrt_init failed. Status = 1
```

Multiworker execution hangs during NCCL init

When your multi-worker execution hangs during NCCL init, you can try to reserve the port used by environment variable `NEURON_RT_ROOT_COMM_ID` by (here we use host:port localhost:48620 as an example but you can use any free port and root node's host IP):

```
sudo sysctl -w net.ipv4.ip_local_reserved_ports=48620
```

Then set the environment variable `NEURON_RT_ROOT_COMM_ID` in your script:

```
os.environ["NEURON_RT_ROOT_COMM_ID"] = "localhost:48620"
```

NRT init error "One or more engines are running. Please restart device by reloading driver"

If you see an error stating "One or more engines are running. Please restart device by reloading driver" please follow the instruction and reload the driver using "`sudo modprobe -r neuron; sudo modprobe neuron;`".

```
2021-Nov-15 20:23:27.0280 3793:3793 ERROR TDRV:tpb_eng_init_hals_v2 CRITICAL HW ERROR:
↪One or more engines are running. Please restart device by reloading driver:
sudo modprobe -r neuron; sudo modprobe neuron;
2021-Nov-15 20:23:27.0280 3793:3793 ERROR TDRV:tdrv_init_one_mla_phase2 nd0 nc0 HAL init
↪failed. error:1
```

NRT error "ERROR TDRV:kbl_model_add Attempting to load an incompatible model!"

If you see an NRT error "ERROR TDRV:kbl_model_add Attempting to load an incompatible model!" this means that the compiler neuronx-cc used to compile the model is too old. See installation instruction to update to latest compiler.

NRT error "ERROR HAL:aws_hal_sprot_config_remap_entry SPROT remap destination address must be aligned size"

If you see an NRT error "ERROR HAL:aws_hal_sprot_config_remap_entry SPROT remap destination address must be aligned size", please check the kernel version and upgrade it to the distribution's latest kernel.

For example, on Ubuntu 18.04.6 LTS, the kernel version 4.15.0-66-generic is known to cause this error when running MLP tutorial. This is due to a known bug in the kernel in aligned memory allocation. To fix this issue, please upgrade your kernel to latest version (i.e. 4.15.0-171-generic):

```
uname -a
sudo apt-get update
sudo  apt-get upgrade
sudo apt-get dist-upgrade
```

Please reboot after the upgrade. Use "uname -a" to check kernel version again after reboot.

NCCL warning : "NCCL WARN Timeout waiting for RX (waited 120 sec) - retrying"

When running multi-worker training, if a graph has collective communication operator like an `all_reduce`, it requires all the workers involved in the collective communication to load the graph in the runtime at approximately same time. If any of the worker doesn't load the graph within a 120 sec window from the first model load by any of the worker, you would see warnings like `NCCL WARN Timeout waiting for RX (waited 120 sec) - retrying`. When you see such warnings check for the following in the log messages:

1. One of the workers is compiling a graph: In multi-worker training, there is a chance that each worker builds a slightly different graph. This would result in cache miss and can result in compilation. Since the compilations during training run are serialized, the first worker can compile and load the graph with collective communication. It would then wait for 120 secs for other works to join. If they don't show up because they are compiling their own graphs, first worker would start throwing a warning message as above. The warning in this case is `non-fatal` and would go away once all workers have compiled their respective graphs and then loaded them. To identify this scenario, look for `No candidate found under ....` logs around the warning. You should also see `.....` which indicates compilation is in progress.

2. Server on one of the nodes crashed: In distributed training across multiple nodes, if the server on one node crashed, the workers on other nodes would keep waiting on model load and you would see above `timeout` logs on those nodes. To identify if the server crashed, check if you see the following error on any of the nodes:

```
`RPC failed with status = "UNAVAILABLE: Socket closed" and grpc_error_string = "{"created
↪":"@1664146011.016500243","description":"Error received from peer ipv4:10.1.24.109:
↪37379","file":"external/com_github_grpc_grpc/src/core/lib/surface/call.cc","file_line":
↪1056,"grpc_message":"Socket closed","grpc_status":14}", maybe retrying the RPC`
```

If you see the above error, then it means there is a server crash and you need to cancel the traning run.

RPC error: "RPC failed with status = 'UNAVAILABLE: Socket closed'"

When you see the above error, it means that the xrt server crashed. When you see such an error, look for the following:

1. Check for any error logs before the `RPC error`. That should indicate the root cause of server crash. Note: The actual error log might be buried because of all the `RPC error` logs that swamp the logs.
2. Sometimes the server can crash because of host OOM. This can happen when we are loading and saving checkpoints. In such cases, you only see `RPC errors` and no other log. You can check if any instance is going out of memory by using tools like dmesg

Error "Assertion `listp->slotinfo[cnt].gen <= GL(dl_tls_generation)' failed" followed by 'RPC failed with status = "UNAVAILABLE: Connection reset by peer"'

The error "Assertion `listp->slotinfo[cnt].gen <= GL(dl_tls_generation)' failed" is intermittent and occurs when using glibc 2.26. To find out the glibc version you have, you can run `ldd --version`. The workaround is to use Ubuntu 20 where glibc is 2.27.

```
INFO: Inconsistency detected by ld.so: ../elf/dl-tls.c: 488: _dl_allocate_tls_init:␣
↪Assertion `listp->slotinfo[cnt].gen <= GL(dl_tls_generation)' failed!
INFO: 2022-10-03 02:16:04.488054: W tensorflow/core/distributed_runtime/rpc/grpc_remote_
↪master.cc:157] RPC failed with status = "UNAVAILABLE: Connection reset by peer" and␣
↪grpc_error_string = "{"created":"@1664763364.487962663","description":"Error received␣
↪from peer ipv4:10.0.9.150:41677","file":"external/com_github_grpc_grpc/src/core/lib/
↪surface/call.cc","file_line":1056,"grpc_message":"Connection reset by peer","grpc_
↪status":14}", maybe retrying the RPC
```

(continues on next page)

(continued from previous page)

RPC connection error: "RPC failed with status = UNAVAILABLE: Connection reset by peer" not preceded by any error

This error may not be preceded by another error like shown in the previous section. In this case, the RPC connection error usually happens when we do distributed training across multiple nodes. When you see such error, please wait for a few minutes. It might be because some node is taking time to setup and hence the other node is not able to connect to it just yet. Once, all nodes are up, training should resume.

Runtime errors "Missing infer_status notification" followed by "inference timeout"

If you get a timeout error like below:

```
ERROR  TDRV:exec_consume_tpb_status_notifications   Missing infer_status notification:␣
↪(end:4)
ERROR  TDRV:exec_consume_infer_status_notifications (FATAL-RT-UNDEFINED-STATE) inference␣
↪timeout (600000 ms) on Neuron Device 4 NC 1, waiting for execution completion␣
↪notification
```

It maybe due to long graph execution time causing synchronization delays exceeding the default timeout. Please try increasing the timeout to larger value using `NEURON_RT_EXEC_TIMEOUT` (unit in seconds) and see if the problem is resolved.

Protobuf Error "TypeError: Descriptors cannot not be created directly."

If you install torch-neuronx after neuronx-cc, you may get the Protobuf error "TypeError: Descriptors cannot not be created directly.". To fix this, please reinstall neuronx-cc using "pip install –force-reinstall neuronx-cc".

```
Traceback (most recent call last):
  File "./run_glue.py", line 570, in <module>
    main()
  File "./run_glue.py", line 478, in main
    data_collator=data_collator,
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪transformers/trainer.py", line 399, in __init__
    callbacks, self.model, self.tokenizer, self.optimizer, self.lr_scheduler
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪transformers/trainer_callback.py", line 292, in __init__
    self.add_callback(cb)
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪transformers/trainer_callback.py", line 309, in add_callback
    cb = callback() if isinstance(callback, type) else callback
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪transformers/integrations.py", line 390, in __init__
    from torch.utils.tensorboard import SummaryWriter  # noqa: F401
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪torch/utils/tensorboard/__init__.py", line 10, in <module>
    from .writer import FileWriter, SummaryWriter  # noqa: F401
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪torch/utils/tensorboard/writer.py", line 9, in <module>
```

(continues on next page)

(continued from previous page)

```
    from tensorboard.compat.proto.event_pb2 import SessionLog
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪tensorboard/compat/proto/event_pb2.py", line 17, in <module>
    from tensorboard.compat.proto import summary_pb2 as tensorboard_dot_compat_dot_proto_
↪dot_summary__pb2
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪tensorboard/compat/proto/summary_pb2.py", line 17, in <module>
    from tensorboard.compat.proto import tensor_pb2 as tensorboard_dot_compat_dot_proto_
↪dot_tensor__pb2
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪tensorboard/compat/proto/tensor_pb2.py", line 16, in <module>
    from tensorboard.compat.proto import resource_handle_pb2 as tensorboard_dot_compat_
↪dot_proto_dot_resource__handle__pb2
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪tensorboard/compat/proto/resource_handle_pb2.py", line 16, in <module>
    from tensorboard.compat.proto import tensor_shape_pb2 as tensorboard_dot_compat_dot_
↪proto_dot_tensor__shape__pb2
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪tensorboard/compat/proto/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File "/home/ec2-user/aws_neuron_venv_pytorch_p37_exp/lib64/python3.7/site-packages/
↪google/protobuf/descriptor.py", line 560, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of date and must be␣
↪regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will use pure-Python␣
↪parsing and will be much slower).
```

TDRV error "Timestamp program stop timeout"

If you see TDRV error "Timestamp program stop timeout", i.e. when rerunning a training script after it was interrupted, try first reloading the driver using `sudo modprobe -r neuron; sudo modprobe neuron;` (make sure neuron-top and/or neuron-monitor are not running).

```
2022-Aug-31 04:59:21.0546 117717:117717 ERROR  TDRV:tsync_wait_eng_stop                 ␣
↪   nd0 nc0 Timestamp program stop timeout (1000 ms)
2022-Aug-31 04:59:21.0546 117717:117717 ERROR  TDRV:tsync_wait_nc_stop                  ␣
↪   nd0 nc0 Error while waiting for timestamp program to end on TPB eng 0
2022-Aug-31 04:59:21.0546 117717:117717 ERROR  TDRV:tsync_timestamps_finish             ␣
↪   nd0 nc0 Failed to stop neuron core
2022-Aug-31 04:59:21.0546 117717:117717 ERROR  TDRV:tdrv_tsync_timestamps               ␣
↪   nd0 nc0 Failed to end timestamp sync programs
2022-Aug-31 04:59:22.0768 117717:117717 ERROR  TDRV:tdrv_destroy                        ␣
↪   TDRV not initialized
2022-Aug-31 04:59:22.0768 117717:117717 ERROR   NRT:nrt_init                            ␣
↪   Failed to initialize devices, error:5
```

Compiler error "module 'numpy' has no attribute 'asscalar'"

When you have a newer version of numpy in the Python environment, compilations may fail with the "error module 'numpy' has no attribute 'asscalar'". Please note the neuronx-cc has the following dependency on numpy "numpy<=1.20.0,>=1.13.3". To workaround this error, please do "pip install –force-reinstall neuronx-cc" to reinstall neuronx-cc with the proper dependencies.

```
ERROR 227874 [neuronx-cc]:␣
↪***************************************************************
ERROR 227874 [neuronx-cc]:  An Internal Compiler Error has occurred
ERROR 227874 [neuronx-cc]:␣
↪***************************************************************
ERROR 227874 [neuronx-cc]:
ERROR 227874 [neuronx-cc]: Error message:  module 'numpy' has no attribute 'asscalar'
ERROR 227874 [neuronx-cc]:
ERROR 227874 [neuronx-cc]: Error class:    AttributeError
ERROR 227874 [neuronx-cc]: Error location: Unknown
ERROR 227874 [neuronx-cc]: Version information:
ERROR 227874 [neuronx-cc]:   NeuronX Compiler version 2.1.0.76+2909d26a2
ERROR 227874 [neuronx-cc]:
ERROR 227874 [neuronx-cc]:   HWM version 2.1.0.7-64eaede08
ERROR 227874 [neuronx-cc]:   NEFF version Dynamic
ERROR 227874 [neuronx-cc]:   TVM not available
ERROR 227874 [neuronx-cc]:   NumPy version 1.23.3
ERROR 227874 [neuronx-cc]:   MXNet not available
ERROR 227874 [neuronx-cc]:
```

Import errors 'generic_type: type "IrValue" is already registered!' or 'generic_type: type "XlaBuilder" is already registered!'

When you encounter a PyTorch import error 'import _XLAC … generic_type: type "IrValue" is already registered!' or 'import _XLAC … generic_type: type "XlaBuilder" is already registered!', please check that TensorFlow and/or JAX are not installed in the Python environment. If they are installed, please uninstall them.

Import error "import _XLAC ImportError: <>/site-packages/_XLAC.cpython-38-x86_64-linux-gnu.so: undefined symbol"

When you encounter a PyTorch import error "import _XLAC ImportError: <>/site-packages/_XLAC.cpython-38-x86_64-linux

1. TensorFlow and/or JAX are not installed in the Python environment. If they are installed, please uninstall them.
2. The installed PyTorch (torch) package major/minor versions match the installed torch-neuronx package's major/minor versions (ie. 1.11). If they don't match, please install the version of PyTorch that matches torch-neuronx.

```
Traceback (most recent call last):
  File "/opt/ml/mlp_train.py", line 11, in <module>
    import torch_xla.core.xla_model as xm
  File "/usr/local/lib/python3.8/site-packages/torch_xla/__init__.py", line 117, in
↪<module>
```

(continues on next page)

(continued from previous page)

```
    import _XLAC
ImportError: /usr/local/lib/python3.8/site-packages/_XLAC.cpython-38-x86_64-linux-gnu.so:
↪ undefined symbol: _ZNK3c10l0TensorImpl7stridesEv
```

NaNs seen with transformers version >= 4.21.0 when running HF BERT fine-tuning or pretraining with XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1

When running HuggingFace BERT (any size) fine-tuning tutorial or pretraining tutorial with transformers version >= 4.21.0 and using XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1, you will see NaNs in the loss immediately at the first step. More details on the issue can be found at pytorch/xla#4152. The workaround is to use 4.20.0 or earlier (the tutorials currently recommend version 4.15.0) or add `transformers.modeling_utils.get_parameter_dtype = lambda x: torch.bfloat16` to the Python script.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

8.16 PyTorch Neuron (`torch-neuronx`) release notes

Table of Contents

- *Release [1.12.0.1.4.0]*
- *Release [1.11.0.1.2.0]*
- *Release [1.11.0.1.1.1]*

PyTorch Neuron for Trainium is a software package that enables PyTorch users to train their models on Trainium.

8.16.1 Release [1.12.0.1.4.0]

Date: 12/12/2022

Summary

What's new in this release

- Added support for PyTorch 1.12.
- Setting XLA_DOWNCAST_BF16=1 now also enables stochastic rounding by default (as done with XLA_USE_BF16=1).
- Added support for *capturing snapshots* of inputs, outputs and graph HLO for debug.
- Fixed issue with parallel compile error when both train and evaluation are enabled in HuggingFace fine-tuning tutorial.
- Added support for LAMB optimizer in FP32 mode.

Resolved Issues

NaNs seen with transformers version >= 4.21.0 when running HF BERT fine-tuning or pretraining with XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1

When running HuggingFace BERT (any size) fine-tuning tutorial or pretraining tutorial with transformers version >= 4.21.0 and using XLA_USE_BF16=1 or XLA_DOWNCAST_BF16=1, you will see NaNs in the loss immediately at the first step. More details on the issue can be found at pytorch/xla#4152. The workaround is to use 4.20.0 or earlier (the tutorials currently recommend version 4.15.0) or add the line `transformers.modeling_utils.get_parameter_dtype = lambda x: torch.bfloat16` to your Python training script (as now done in latest tutorials). A permanent fix will become part of an upcoming HuggingFace transformers release.

Known Issues and Limitations

Convolution is not supported

In this release, convolution is not supported.

Number of data parallel training workers on one Trn1 instance

The number of workers used in single-instance data parallel training can be one of the following values: 1 or 2 for trn1.2xlarge and 1, 2, 8 or 32 for trn1.32xlarge.

8.16.2 Release [1.11.0.1.2.0]

Date: 10/27/2022

Summary

What's new in this release

- Added support for argmax.
- Clarified error messages for runtime errors `NRT_UNINITIALIZED` and `NRT_CLOSED`.
- When multi-worker training is launched using torchrun on one instance, framework now handles runtime state cleanup at end of training.

Resolved Issues

Drop-out rate ignored in dropout operation

A known issue in the compiler's implementation of dropout caused drop-rate to be ignored in the last release. It is fixed in the current release.

Runtime error "invalid offset in Coalesced_memloc_…" followed by "Failed to process dma block: 1703"

Previously, when running MRPC fine-tuning tutorial with `bert-base-*` model, you would encounter runtime error "invalid offset in Coalesced_memloc_…" followed by "Failed to process dma block: 1703". This is fixed in the current release.

Compilation error: "TongaSBTensor[0x7fb2a46e0830]:TongaSB partitions[0] uint8 %138392[128, 512]"

Previously, when compiling MRPC fine-tuning tutorial with `bert-large-*` and FP32 (no XLA_USE_BF16=1) for two workers or more, you would encounter compiler error that looks like `Error message: TongaSBTensor[0x7fb2a46e0830]:TongaSB partitions[0] uint8 %138392[128, 512]` followed by `Error class: KeyError`. Single worker fine-tuning is not affected. This is fixed in the current release.

Known Issues and Limitations

Convolution is not supported

In this release, convolution is not supported.

Number of data parallel training workers on one Trn1 instance

The number of workers used in single-instance data parallel training can be one of the following values: 1 or 2 for trn1.2xlarge and 1, 2, 8 or 32 for trn1.32xlarge.

8.16.3 Release [1.11.0.1.1.1]

Date: 10/10/2022

Summary

This is the initial release of PyTorch Neuron that supports Trainium for users to train their models on the new EC2 Trn1 instances.

What's new in this release

Announcing the first PyTorch Neuron release for training.

- XLA device support for Trainium
- PyTorch 1.11 with XLA backend support in torch.distributed
- torch-xla distributed support
- Single-instance and multi-instance distributed training using torchrun
- Support for ParallelCluster and SLURM with node-level scheduling granularity
- Persistent cache for compiled graph
- *neuron_parallel_compile* utility to help speed up compilation

- Optimizer support: SGD, AdamW
- Loss functions supported: NLLLoss
- Python versions supported: 3.7, 3.8
- Multi-instance training support with EFA
- Support PyTorch's BF16 automatic mixed precision

Known Issues and Limitations

Convolution is not supported

In this release, convolution is not supported.

Number of data parallel training workers on one Trn1 instance

The number of workers used in single-instance data parallel training can be one of the following values: 1 or 2 for trn1.2xlarge and 1, 2, 8 or 32 for trn1.32xlarge.

Drop-out rate ignored in dropout operation

A known issue in the compiler's implementation of dropout caused drop-rate to be ignored. Will be fixed in a follow-on release.

Runtime error "invalid offset in Coalesced_memloc_…" followed by "Failed to process dma block: 1703"

Currently, when running MRPC fine-tuning tutorial with `bert-base-*` model, you will encounter runtime error "invalid offset in Coalesced_memloc_…" followed by "Failed to process dma block: 1703". This issue will be fixed in an upcoming release.

Compilation error: "TongaSBTensor[0x7fb2a46e0830]:TongaSB partitions[0] uint8 %138392[128, 512]"

When compiling MRPC fine-tuning tutorial with `bert-large-*` and FP32 (no XLA_USE_BF16=1) for two workers or more, you will encounter compiler error that looks like `Error message: TongaSBTensor[0x7fb2a46e0830]: TongaSB partitions[0] uint8 %138392[128, 512]` followed by `Error class: KeyError`. Single worker fine-tuning is not affected. This issue will be fixed in an upcoming release.

This document is relevant for: `Trn1`

Inference

Setup Guide for Inf1

This document is relevant for: `Inf1`

8.17 Install PyTorch Neuron (`torch-neuron`)

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

8.17.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate
```

PyTorch 1.11.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
```

(continues on next page)

(continued from previous page)

```
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.10.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.9.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
```

(continues on next page)

(continued from previous page)

```
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.8.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
```

(continues on next page)

(continued from previous page)

```
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

PyTorch 1.7.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

8.17.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.12.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate PyTorch
source activate
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate PyTorch
source activate
```

PyTorch 1.11.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.10.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.9.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.8.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

PyTorch 1.7.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

8.17.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate
```

PyTorch 1.11.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.* torchvision
```

PyTorch 1.10.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################
#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.* torchvision
```

PyTorch 1.9.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.* torchvision
```

PyTorch 1.8.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.* torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##################################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

PyTorch 1.7.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* torchvision
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* torchvision
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* torchvision
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.* torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.18 Update to latest PyTorch Neuron (`torch-neuron`)

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

8.18.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-tools -y

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##############################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

8.18.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron neuron-cc[tensorflow] "protobuf==3.20.1" torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* neuron-cc[tensorflow] "protobuf==3.20.1"␣
↪torchvision
```

8.18.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install --upgrade torch-neuron torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.11.0.* torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.10.2.* torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.9.1.* torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##############################################################################

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.8.1.* torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source pytorch_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron PyTorch
pip install --upgrade torch-neuron==1.7.1.* torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19 Install previous PyTorch Neuron releases (`torch-neuron`)

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

This document is relevant for: `Inf1`

8.19.1 Install PyTorch Neuron (Neuron 2.5.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.0 torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-tools=2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
yum remove' before installing the older package
sudo yum install aws-neuronx-tools-2.5.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.
↪5.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.13.5.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.12.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.12.1.2.5.8.0 torchvision
```

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##############################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##############################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.5.8.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.10.2.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.5.8.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.5.8.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
```

(continues on next page)

(continued from previous page)

```
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.5.8.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.33.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.33.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.5.8.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.2 Install PyTorch Neuron (Neuron 2.4.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
```

(continues on next page)

(continued from previous page)

```
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
```

(continues on next page)

(continued from previous page)

```
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.5.16.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
```

(continues on next page)

(continued from previous page)

```
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#####################################################
#   Only for Ubuntu 20 - Install Python3.7
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.6.5.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.6.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.3 Install PyTorch Neuron (Neuron 2.3.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
```

(continues on next page)

(continued from previous page)

```
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
```

(continues on next page)

(continued from previous page)

```
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
```

(continues on next page)

(continued from previous page)

```
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to 'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪7.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.7.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##############################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##############################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##############################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################################
↪#######################

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuronx-dkms=2.5.41.0 --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms-2.5.41.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.4 Install PyTorch Neuron (Neuron 1.19.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

(continues on next page)

(continued from previous page)

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.
↪4.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.11.4.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.11.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.11.0.2.3.0.0 torchvision
```

PyTorch 1.10.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.2.2.3.0.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.3.0.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.3.0.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.3.0.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.5 Install PyTorch Neuron (Neuron 1.18.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.
↪3.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.10.3.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.2.0.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
```

(continues on next page)

(continued from previous page)

```
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.2.0.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.2.0.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.7.1.2.2.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.2.0.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.2.0.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.6 Install PyTorch Neuron (Neuron 1.17.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

(continues on next page)

(continued from previous page)

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.9.1.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.10.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.10.1.2.1.7.0 torchvision
```

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
```

(continues on next page)

(continued from previous page)

```
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.1.7.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.1.7.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.7.1.2.1.7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.1.7.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.1.7.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.7 Install PyTorch Neuron (Neuron 1.16.3)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

(continues on next page)

(continued from previous page)

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪5.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.536.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.536.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.536.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.536.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.8 Install PyTorch Neuron (Neuron 1.16.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

(continues on next page)

(continued from previous page)

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#####################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.8.
↪2.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.468.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.468.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.468.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.468.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.9 Install PyTorch Neuron (Neuron 1.16.1)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

(continues on next page)

(continued from previous page)

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.327.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#####################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.7.
↪3.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.9.1.2.0.392.0 torchvision
```

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.2.0.392.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.2.0.392.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.6.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.6.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.2.0.392.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.10 Install PyTorch Neuron (Neuron 1.15.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
```

(continues on next page)

(continued from previous page)

```
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.11 Install PyTorch Neuron (Neuron 1.15.1)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
```

(continues on next page)

(continued from previous page)

```
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.12 Install PyTorch Neuron (Neuron 1.15.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
```

(continues on next page)

(continued from previous page)

```
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.6.
↪13.0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.21.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.21.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
###################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

###################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
###################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.21.0 torchvision
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.19.13 Install PyTorch Neuron (Neuron 1.14.2)

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
```

(continues on next page)

(continued from previous page)

```
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name pytorch_venv --display-name "Python (Neuron␣
↪PyTorch)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 neuron-cc[tensorflow] "protobuf==3.20.1"==1.5.5.
↪0 torchvision
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

PyTorch 1.8.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.8.1.1.5.12.0 torchvision
```

PyTorch 1.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.7.1.1.5.12.0 torchvision
```

PyTorch 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 torchvision
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv pytorch_venv
source pytorch_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 torchvision
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 torchvision
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate PyTorch
```

(continues on next page)

(continued from previous page)

```
source activate aws_neuron_pytorch_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron PyTorch
pip install torch-neuron==1.5.1.1.5.12.0 torchvision
```

This document is relevant for: Inf1

This document is relevant for: Inf1

This document is relevant for: Inf1

8.20 Install with support for cxx11 ABI

Warning: The intended user of this guide is using a custom built version of `torch` or compiling a non-python application which must be built using the cxx11 ABI.

Most applications do not require this specialized distribution.

For regular installation instructions see: *Fresh install*

The standard `torch-neuron` packages (which are normally installed according to the *Fresh install* guide) are compiled with the pre-cxx11 ABI and linked against the pre-cxx11 `libtorch`. These compilation options ensure that the `torch-neuron` ABI matches the *publicly* released version of the `torch` package that is installed from the default PyPI index.

To support applications with specific ABI requirements, Neuron distributes packages which are linked against the cxx11 version of `libtorch`. These `torch-neuron` packages are built using the `-D_GLIBCXX_USE_CXX11_ABI=1` compilation flag.

The only difference between these packages and the standard packages is the torch plugin library contained within the package. This is the `libtorchneuron.so` library located in the `torch_neuron/lib/` package directory. All other libraries and python files within the packages are identical. This means that these cxx11-compatible packages are drop-in replacements in environments that are incompatible with the standard releases of `torch-neuron`. Behavior is identical whether compiling models or executing inferences.

8.20.1 Installation

All versions of the library are available to download from the following pip index:

```
https://pip.repos.neuron.amazonaws.com/cxx11
```

To install a wheel, it is recommended to use the `--no-deps` flag since versions of `torch` compiled using the cxx11 ABI are not distributed on this index.

```
pip install --extra-index-url=https://pip.repos.neuron.amazonaws.com/cxx11 torch-neuron -
↪-no-deps
```

Specific versions of `torch-neuron` with cxx11 ABI support can be installed just like standard versions of `torch-neuron`.

```
pip install --extra-index-url=https://pip.repos.neuron.amazonaws.com/cxx11 "torch-neuron>
↪=1.8" --no-deps
pip install --extra-index-url=https://pip.repos.neuron.amazonaws.com/cxx11 "torch-
↪neuron==1.9.1" --no-deps
pip install --extra-index-url=https://pip.repos.neuron.amazonaws.com/cxx11 "torch-neuron
↪<1.10" --no-deps
```

Important: This pip index does not include a distribution of `torch` compiled with the new cxx11 ABI. The intent of this index is *only* to provide Neuron SDK wheels.

The version of `torch` that is distributed on the default PyPI index is compiled with the old pre-cxx11 ABI.

If a cxx11 `torch-neuron` package is installed *with* dependencies using the *default* PyPI index, then the installed version of `torch` will be using the pre-cxx11 ABI and `torch-neuron` will be using the cxx11 ABI. This ABI mismatch will lead to errors in both python usage and at link time for non-python applications.

8.20.2 FAQ

When should I use a cxx11 torch-neuron wheel?

Distributions compiled with the new cxx11 ABI should only be used in the following cases:

1. You have built your own version of `torch` which uses the new cxx11 ABI and need a corresponding version of `torch-neuron` that is compatible.
2. You are compiling an application against a `libtorch` which uses the cxx11 ABI and would like to include `libtorchneuron.so` as well. Torch distributes these cxx11 `libtorch` libraries with a `libtorch-cxx11` prefix.

   Example:

   ```
   https://download.pytorch.org/libtorch/cpu/libtorch-cxx11-abi-shared-with-
   ↪deps-1.10.2%2Bcpu.zip
   ```

Can I download a library/header zip file similar to the torch distribution?

Currently `torch-neuron` does not distribute a bundled library `.zip` with only library/header files.

The recommended alternative when compiling `libtorchneuron.so` into a non-python application is to install the `torch-neuron` wheel using `pip` according to the installation instructions. Then use the `libtorchneuron.so` library from within the python `site-packages` directory.

A second alternative to isolate the package contents from a python environment is to download the wheel and unpack the contents:

```
pip download --extra-index-url=https://pip.repos.neuron.amazonaws.com/cxx11 torch-neuron␣
↪--no-deps
wheel unpack torch_neuron-*.whl
```

If the exact version of the `torch-neuron` package is known and no python/pip is available in the build environment, an alternative to is fetch the package file directly and `unzip` the wheel:

```
wget https://pip.repos.neuron.amazonaws.com/cxx11/torch-neuron/torch_neuron-<VERSION>-
↪py3-none-any.whl
unzip torch_neuron-<VERSION>-py3-none-any.whl
```

How can I know which ABI torch-neuron is using?

Packages which use the pre-cxx11 ABI have no local identifier and use the following version scheme:

```
<torch version>.<neuron version>
```

Packages which use the cxx11 ABI have a `+cxx11` local identifier and use following version scheme:

```
<torch version>.<neuron version>+cxx11
```

This allows the ABI to be validated in the by inspecting the local identifier (or version suffix).

Example:

```
1.8.1.0.0.0.0+cxx11
1.9.1.0.0.0.0+cxx11
1.10.2.0.0.0.0+cxx11
```

How can I know which ABI torch is using?

The `torch` python package provides an API at the that allows you to check if the underlying `libtorch` was compiled with the cxx11 ABI:

```
import torch
torch.compiled_with_cxx11_abi()  # True/False
```

Currently `torch-neuron` does not have an equivalent API. If the cxx11 ABI was used, it will be visible in the version string (See *How can I know which ABI torch-neuron is using?*).

8.20.3 Troubleshooting

What python errors could I see if I mix ABI versions?

Using a version of `torch` compiled with the cxx11 ABI will trigger an error in the python interpreter when importing a version of `torch-neuron` using the old (pre-cxx11) ABI from the standard index. This will manifest as an error when the `import torch_neuron` statement is executed.

```
Traceback (most recent call last):
  File "/python3.7/site-packages/torch_neuron/__init__.py", line 64, in <module>
    _register_extension()
  File "/python3.7/site-packages/torch_neuron/__init__.py", line 60, in _register_
↪extension
    torch.ops.load_library(neuron_op_filename)
  File "/python3.7/site-packages/torch/_ops.py", line 110, in load_library
    ctypes.CDLL(path)
  File "/python3.7/ctypes/__init__.py", line 364, in __init__
```

(continues on next page)

(continued from previous page)

```
    self._handle = _dlopen(self._name, mode)
OSError: /python3.7/site-packages/torch_neuron/lib/libtorchneuron.so: undefined symbol: _
↪ZN5torch6detail10class_baseC2ERKSsS3_SsRKSt9type_infoS6_
```

Similarly if using the standard pre-cxx11 version of `torch` with the cxx11 version of `torch-neuron` will also cause an error upon import.

```
Traceback (most recent call last):
  File "/python3.7/site-packages/torch_neuron/__init__.py", line 79, in <module>
    _register_extension()
  File "/python3.7/site-packages/torch_neuron/__init__.py", line 75, in _register_
↪extension
    torch.ops.load_library(neuron_op_filename)
  File "/python3.7/site-packages/torch/_ops.py", line 110, in load_library
    ctypes.CDLL(path)
  File "/python3.7/ctypes/__init__.py", line 364, in __init__
    self._handle = _dlopen(self._name, mode)
OSError: /python3.7/site-packages/torch_neuron/lib/libtorchneuron.so: undefined symbol: _
↪ZN5torch3jit17parseSchemaOrNameERKNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE
```

In either of these cases, the remedy is to ensure that the ABI of the `torch` distribution matches the ABI of the `torch-neuron` distribution.

What compiler/linking errors could I see if I mix ABI versions?

If you link an application which uses the old (pre-cxx11) ABI `libtorchneuron.so` with a cxx11 version of `torch`, this will trigger a link error.

```
libtorchneuron.so: undefined reference to `torch::detail::class_base::class_base(std::
↪string const&, std::string const&, std::string, std::type_info const&, std::type_info␣
↪const&)'
libtorchneuron.so: undefined reference to `c10::Error::Error(c10::SourceLocation, std::
↪string)'
libtorchneuron.so: undefined reference to `c10::detail::torchInternalAssertFail(char␣
↪const*, char const*, unsigned int, char const*, std::string const&)'
libtorchneuron.so: undefined reference to `c10::ClassType::getMethod(std::string const&)␣
↪const'
libtorchneuron.so: undefined reference to `c10::ivalue::ConstantString::create(std::
↪string)'
libtorchneuron.so: undefined reference to `c10::DeviceTypeName(c10::DeviceType, bool)'
libtorchneuron.so: undefined reference to `torch::jit::parseSchema(std::string const&)'
libtorchneuron.so: undefined reference to `unsigned short caffe2::TypeMeta::_typeMetaData
↪<std::string>()'
libtorchneuron.so: undefined reference to `c10::Warning::warn(c10::SourceLocation const&,
↪ std::string const&, bool)'
libtorchneuron.so: undefined reference to `torch::jit::parseSchemaOrName(std::string␣
↪const&)'
libtorchneuron.so: undefined reference to `c10::Symbol::fromQualString(std::string const&
↪)'
libtorchneuron.so: undefined reference to `c10::Error::Error(std::string, std::string,␣
↪void const*)'
libtorchneuron.so: undefined reference to `c10::detail::infer_schema::make_function_
↪schema(std::string&&, std::string&&, c10::ArrayRef<c10::detail::infer_schema::
↪ArgumentDef>, c10::ArrayRef<c10::detail::infer_schema::ArgumentDef>)'
```

(continues on next page)

(continued from previous page)

```
libtorchneuron.so: undefined reference to `c10::detail::torchCheckFail(char const*, char␣
↪const*, unsigned int, std::string const&)'
libtorchneuron.so: undefined reference to `torch::jit::canonicalSchemaString(c10::
↪FunctionSchema const&)'
```

Similarly, an error will also occur in the opposite scenario where the cxx11 `libtorchneuron.so` library is used with the pre-cxx11 `libtorch`:

```
libtorchneuron.so: undefined reference to `c10::ivalue::ConstantString::create(std::__
↪cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >)'
libtorchneuron.so: undefined reference to `torch::jit::parseSchemaOrName(std::__cxx11::
↪basic_string<char, std::char_traits<char>, std::allocator<char> > const&)'
libtorchneuron.so: undefined reference to `c10::Error::Error(c10::SourceLocation, std::__
↪cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >)'
libtorchneuron.so: undefined reference to `c10::Error::Error(std::__cxx11::basic_string
↪<char, std::char_traits<char>, std::allocator<char> >, std::__cxx11::basic_string<char,
↪ std::char_traits<char>, std::allocator<char> >, void const*)'
libtorchneuron.so: undefined reference to `torch::jit::canonicalSchemaString[abi:
↪cxx11](c10::FunctionSchema const&)'
libtorchneuron.so: undefined reference to `torch::detail::class_base::class_base(std::__
↪cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::_
↪_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::
↪__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::type_
↪info const&, std::type_info const&)'
libtorchneuron.so: undefined reference to `c10::detail::torchInternalAssertFail(char␣
↪const*, char const*, unsigned int, char const*, std::__cxx11::basic_string<char, std::
↪char_traits<char>, std::allocator<char> > const&)'
libtorchneuron.so: undefined reference to `c10::detail::torchCheckFail(char const*, char␣
↪const*, unsigned int, std::__cxx11::basic_string<char, std::char_traits<char>, std::
↪allocator<char> > const&)'
libtorchneuron.so: undefined reference to `c10::detail::infer_schema::make_function_
↪schema(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&
↪&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&&,␣
↪c10::ArrayRef<c10::detail::infer_schema::ArgumentDef>, c10::ArrayRef<c10::detail::
↪infer_schema::ArgumentDef>)'
libtorchneuron.so: undefined reference to `torch::jit::parseSchema(std::__cxx11::basic_
↪string<char, std::char_traits<char>, std::allocator<char> > const&)'
libtorchneuron.so: undefined reference to `c10::DeviceTypeName[abi:cxx11](c10::
↪DeviceType, bool)'
libtorchneuron.so: undefined reference to `c10::Symbol::fromQualString(std::__cxx11::
↪basic_string<char, std::char_traits<char>, std::allocator<char> > const&)'
libtorchneuron.so: undefined reference to `unsigned short caffe2::TypeMeta::_typeMetaData
↪<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >()'
libtorchneuron.so: undefined reference to `c10::ClassType::getMethod(std::__cxx11::basic_
↪string<char, std::char_traits<char>, std::allocator<char> > const&) const'
libtorchneuron.so: undefined reference to `c10::Warning::warn(c10::SourceLocation const&,
↪ std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&
↪, bool)'
```

In either of these cases, the remedy is to ensure that the ABI of the `libtorch` distribution matches the ABI of the `libtorchneuron.so` distribution.

The `torch` ABI must match the `torch-neuron` ABI or an error will occur.

This document is relevant for: `Inf1`

Tutorials (`torch-neuron`)

Computer Vision Tutorials

- ResNet-50 tutorial *[html]* [notebook]
- PyTorch YOLOv4 tutorial *[html]* [notebook]

8.21 ResNet50 model for Inferentia

8.21.1 Introduction:

In this tutorial we will compile and deploy a ResNet50 model for inference on Inferentia.

This Jupyter notebook should run on an inf1.6xlarge instance. The inference part of this tutorial requires an inf1 instance, not the compilation stage. For simplicity we will run this tutorial on an inf1.6xlarge, but in real life scenarios the compilation should be done on a compute instance and the deployment on an inf1 instance to save costs.

In this tutorial we provide three main sections:

1. Compile the ResNet50 model and infer with a batch size of 1
2. Run the same compiled model on multiple NeuronCores using `torch.neuron.DataParallel` and dynamic batching
3. Compile the ResNet50 model with a batch size of 5 and run it on multiple NeuronCores using `torch.neuron.DataParallel` for optimal performance on Inferentia

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

8.21.2 Install Dependencies:

This tutorial requires the following pip packages:

- `torch>=1.8`
- `torch-neuron`
- `torchvision`
- `neuron-cc[tensorflow]`

These will be installed by default when configuring your environment using the Neuron PyTorch setup guide.

8.21.3 Compile model for Neuron

The following step will compile the ResNet50 model for Inferentia. This will take a few minutes. At the end of script execution, the compiled model is saved as `resnet50_neuron.pt` in your local directory

[ ]:
```
import torch
from torchvision import models, transforms, datasets
import torch_neuron

# Create an example input for compilation
image = torch.zeros([1, 3, 224, 224], dtype=torch.float32)

# Load a pretrained ResNet50 model
model = models.resnet50(pretrained=True)

# Tell the model we are using it for evaluation (not training)
model.eval()

# Analyze the model - this will show operator support and operator count
torch.neuron.analyze_model(model, example_inputs=[image])

# Compile the model using torch.neuron.trace to create a Neuron model
# that that is optimized for the Inferentia hardware
model_neuron = torch.neuron.trace(model, example_inputs=[image])

# The output of the compilation step will report the percentage of operators that
# are compiled to Neuron, for example:
#
# INFO:Neuron:The neuron partitioner created 1 sub-graphs
# INFO:Neuron:Neuron successfully compiled 1 sub-graphs, Total fused subgraphs = 1,␣
↪Percent of model sub-graphs successfully compiled = 100.0%
#
# We will also be warned if there are operators that are not placed on the Inferentia␣
↪hardware

# Save the compiled model
model_neuron.save("resnet50_neuron.pt")
```

8.21.4 Run inference on Inferentia

We can use the compiled Neuron model to run inference on Inferenita.

In the following example, we preprocess a sample image for inference using the CPU model and Neuron model. We compare the predicted labels from the CPU model and Neuron model to verify that they are the same.

Important: Do not perform inference with a Neuron traced model on a non-Neuron supported instance, as the results will not be calculated properly.

Define a preprocessing function

We define a basic image preprocessing function that loads a sample image and labels, normalizes and batches the image, and transforms the image into a tensor for inference using the compiled Neuron model.

[ ]:
```
import json
import os
from urllib import request

# Create an image directory containing a sample image of a small kitten
os.makedirs("./torch_neuron_test/images", exist_ok=True)
request.urlretrieve("https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/
↪docs/images/kitten_small.jpg",
                    "./torch_neuron_test/images/kitten_small.jpg")

# Fetch labels to output the top classifications
request.urlretrieve("https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_
↪class_index.json","imagenet_class_index.json")
idx2label = []

# Read the labels and create a list to hold them for classification
with open("imagenet_class_index.json", "r") as read_file:
    class_idx = json.load(read_file)
    idx2label = [class_idx[str(k)][1] for k in range(len(class_idx))]
```

[ ]:
```
import numpy as np

def preprocess(batch_size=1, num_neuron_cores=1):
    # Define a normalization function using the ImageNet mean and standard deviation
    normalize = transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225])

    # Resize the sample image to [1, 3, 224, 224], normalize it, and turn it into a␣
↪tensor
    eval_dataset = datasets.ImageFolder(
        os.path.dirname("./torch_neuron_test/"),
        transforms.Compose([
        transforms.Resize([224, 224]),
        transforms.ToTensor(),
        normalize,
        ])
    )
    image, _ = eval_dataset[0]
    image = torch.tensor(image.numpy()[np.newaxis, ...])

    # Create a "batched" image with enough images to go on each of the available␣
↪NeuronCores
    # batch_size is the per-core batch size
    # num_neuron_cores is the number of NeuronCores being used
    batch_image = image
    for i in range(batch_size * num_neuron_cores - 1):
        batch_image = torch.cat([batch_image, image], 0)
```

(continues on next page)

(continued from previous page)

```
    return batch_image
```

Run inference using the Neuron model

We import the necessary python modules, load the torch-neuron compiled model, and run inference on Inferentia.

By default, the Neuron model will run on a single NeuronCore. In the next section, we will see how to run the Neuron model on multiple NeuronCores to fully saturate our hardware for optimal performance on Inferentia.

[ ]:
```
import torch
from torchvision import models, transforms, datasets
import torch_neuron

# Get a sample image
image = preprocess()

# Run inference using the CPU model
output_cpu = model(image)

# Load the compiled Neuron model
model_neuron = torch.jit.load('resnet50_neuron.pt')

# Run inference using the Neuron model
output_neuron = model_neuron(image)

# Verify that the CPU and Neuron predictions are the same by comparing
# the top-5 results
top5_cpu = output_cpu[0].sort()[1][-5:]
top5_neuron = output_neuron[0].sort()[1][-5:]

# Lookup and print the top-5 labels
top5_labels_cpu = [idx2label[idx] for idx in top5_cpu]
top5_labels_neuron = [idx2label[idx] for idx in top5_neuron]
print("CPU top-5 labels: {}".format(top5_labels_cpu))
print("Neuron top-5 labels: {}".format(top5_labels_neuron))
```

8.21.5 Run Inference using torch.neuron.DataParallel

To fully leverage the Inferentia hardware we want to use all avaialable NeuronCores. An inf1.xlarge and inf1.2xlarge have four NeuronCores, an inf1.6xlarge has 16 NeuronCores, and an inf1.24xlarge has 64 NeuronCores. For maximum performance on Inferentia hardware, we can use `torch.neuron.DataParallel` to utilize all available NeuronCores.

`torch.neuron.DataParallel` implements data parallelism at the module level by duplicating the Neuron model on all available NeuronCores and distributing data across the different cores for parallelized inference.

In the following section, we will run inference using the `torch.neuron.DataParallel` module to fully saturate the Inferentia hardware. We benchmark the model to collect throughput and latency statistics.

Note: `torch.neuron.DataParallel` is new with Neuron 1.16.0. Please ensure you are using the latest Neuron package to run the following sections.

Define a benchmarking function

We create a function that handles benchmarking the Neuron model to collect throughput and latency metrics.

[ ]:
```
from time import time

def benchmark(model, image):
    print('Input image shape is {}'.format(list(image.shape)))

    # The first inference loads the model so exclude it from timing
    results = model(image)

    # Collect throughput and latency metrics
    latency = []
    throughput = []

    # Run inference for 100 iterations and calculate metrics
    num_infers = 100
    for _ in range(num_infers):
        delta_start = time()
        results = model(image)
        delta = time() - delta_start
        latency.append(delta)
        throughput.append(image.size(0)/delta)

    # Calculate and print the model throughput and latency
    print("Avg. Throughput: {:.0f}, Max Throughput: {:.0f}".format(np.mean(throughput),␣
↪np.max(throughput)))
    print("Latency P50: {:.0f}".format(np.percentile(latency, 50)*1000.0))
    print("Latency P90: {:.0f}".format(np.percentile(latency, 90)*1000.0))
    print("Latency P95: {:.0f}".format(np.percentile(latency, 95)*1000.0))
    print("Latency P99: {:.0f}\n".format(np.percentile(latency, 99)*1000.0))
```

Run Inference using torch.neuron.DataParallel

We create the `torch.neuron.DataParallel` module using the compiled Neuron model, get a sample image, and benchmark the parallelized model on Neuron.

[ ]:
```
# Create a torch.neuron.DataParallel module using the compiled Neuron model
# By default, torch.neuron.DataParallel will use four cores on an inf1.xlarge
# or inf1.2xlarge, 16 cores on an inf1.6xlarge, and 24 cores on an inf1.24xlarge
model_neuron_parallel = torch.neuron.DataParallel(model_neuron)

# Get sample image with batch size=1 per NeuronCore
batch_size = 1

# For an inf1.xlarge or inf1.2xlarge, set num_neuron_cores = 4
num_neuron_cores = 16

image = preprocess(batch_size=batch_size, num_neuron_cores=num_neuron_cores)

# Benchmark the model
benchmark(model_neuron_parallel, image)
```

8.21.6 Run inference with dynamic batch sizes

Batch size has a direct impact on model performance. The Inferentia chip is optimized to run with small batch sizes. This means that a Neuron compiled model can outperform a GPU model, even if running single digit batch sizes.

As a general best practice, we recommend optimizing your model's throughput by compiling the model with a small batch size and gradually increasing it to find the peak throughput on Inferentia.

Dynamic batching is a feature that allows you to use tensor batch sizes that the Neuron model was not originally compiled against. This is necessary because the underlying Inferentia hardware will always execute inferences with the batch size used during compilation. Fixed batch size execution allows tuning the input batch size for optimal performance. For example, batch size 1 may be best suited for an ultra-low latency on-demand inference application, while batch size > 1 can be used to maximize throughput for offline inferencing. Dynamic batching is implemented by slicing large input tensors into chunks that match the batch size used during the `torch.neuron.trace` compilation call.

The `torch.neuron.DataParallel` class automatically enables dynamic batching on eligible models. This allows us to run inference in applications that have inputs with a variable batch size without needing to recompile the model.

In the following example, we use the same `torch.neuron.DataParallel` module to run inference using several different batch sizes. Notice that latency increases consistently as the batch size increases. Throughput increases as well, up until a certain point where the input size becomes too large to be efficient.

[ ]:
```
# using the same DataParallel model_neuron_parallel model, we can run
# inference on inputs with a variable batch size without recompiling
batch_sizes = [2, 3, 4, 5, 6, 7]
for batch_size in batch_sizes:
    print('Batch size: {}'.format(batch_size))
    image = preprocess(batch_size=batch_size, num_neuron_cores=num_neuron_cores)

    # Benchmark the model for each input batch size
    benchmark(model_neuron_parallel, image)
```

8.21.7 Compile and Infer with different batch sizes on multiple NeuronCores

Dynamic batching using small batch sizes can result in sub-optimal throughput because it involves slicing tensors into chunks and iteratively sending data to the hardware. Using a larger batch size at compilation time can use the Inferentia hardware more efficiently in order to maximize throughput. You can test the tradeoff between individual request latency and total throughput by fine-tuning the input batch size.

In the following example, we recompile our model using a batch size of 5 and run the model using `torch.neuron.DataParallel` to fully saturate our Inferentia hardware for optimal performance.

[ ]:
```
# Create an input with batch size 5 for compilation
batch_size = 5
image = torch.zeros([batch_size, 3, 224, 224], dtype=torch.float32)

# Recompile the ResNet50 model for inference with batch size 5
model_neuron = torch.neuron.trace(model, example_inputs=[image])

# Export to saved model
model_neuron.save("resnet50_neuron_b{}.pt".format(batch_size))
```

Run inference with batch size of 5 using the Neuron model compiled for a batch size of 5.

[ ]:
```
batch_size = 5

# Load compiled Neuron model
model_neuron = torch.jit.load("resnet50_neuron_b{}.pt".format(batch_size))

# Create DataParallel model
model_neuron_parallel = torch.neuron.DataParallel(model_neuron)

# Get sample image with batch size=5
image = preprocess(batch_size=batch_size, num_neuron_cores=num_neuron_cores)

# Benchmark the model
benchmark(model_neuron_parallel, image)
```

You can experiment with different batch size values to see what gives the best overall throughput on Inferentia.

8.22 Evaluate YOLO v4 on Inferentia

8.22.1 Introduction

This tutorial walks through compiling and evaluating YOLO v4 model implemented in PyTorch on Inferentia.

The tutorial has five main sections:

1. Define YOLO v4 model in PyTorch
2. Download the COCO 2017 evaluation dataset and define the data loader function
3. Build, Compile, and Save Neuron-Optimized YOLO v4 TorchScript
4. Evaluate Accuracy on the COCO 2017 Dataset
5. Benchmark COCO Dataset Performance of the Neuron-Optimized TorchScript

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

8.22.2 Install Dependencies:

This tutorial requires the following pip packages:

- `torch-neuron`
- `torchvision`
- `pillow`
- `pycocotools`
- `neuron-cc[tensorflow]`

Many of these packages will be installed by default when configuring your environment using the Neuron PyTorch setup guide. The additional dependencies must be installed here.

[ ]:
```
!pip install --upgrade pillow pycocotools
```

8.22.3 Part 1: Define YOLO v4 model in PyTorch

The following PyTorch model definition is from https://github.com/Tianxiaomo/pytorch-YOLOv4/.

[ ]:
```
import numpy as np
import torch
import torch.neuron
from torch import nn
import torch.nn.functional as F
import os
import warnings

# Setting up NeuronCore groups for inf1.6xlarge with 16 cores
n_cores = 16 # This value should be 4 on inf1.xlarge and inf1.2xlarge
os.environ['NEURON_RT_NUM_CORES'] = str(n_cores)


class Mish(torch.nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        x = x * (torch.tanh(torch.nn.functional.softplus(x)))
        return x


class Upsample(nn.Module):
    def __init__(self):
        super(Upsample, self).__init__()

    def forward(self, x, target_size, inference=False):
        assert (x.data.dim() == 4)

        if inference:

            return x.view(x.size(0), x.size(1), x.size(2), 1, x.size(3), 1).\
                    expand(x.size(0), x.size(1), x.size(2), target_size[2] // x.size(2),␣
↪x.size(3), target_size[3] // x.size(3)).\
                    contiguous().view(x.size(0), x.size(1), target_size[2], target_
↪size[3])
        else:
            return F.interpolate(x, size=(target_size[2], target_size[3]), mode='nearest
↪')


class Conv_Bn_Activation(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, stride, activation,␣
↪bn=True, bias=False):
        super().__init__()
        pad = (kernel_size - 1) // 2

        self.conv = nn.ModuleList()
        if bias:
```

(continues on next page)

(continued from previous page)

```
            self.conv.append(nn.Conv2d(in_channels, out_channels, kernel_size, stride,␣
↪pad))
        else:
            self.conv.append(nn.Conv2d(in_channels, out_channels, kernel_size, stride,␣
↪pad, bias=False))
        if bn:
            self.conv.append(nn.BatchNorm2d(out_channels))
        if activation == "mish":
            self.conv.append(Mish())
        elif activation == "relu":
            self.conv.append(nn.ReLU(inplace=True))
        elif activation == "leaky":
            self.conv.append(nn.LeakyReLU(0.1, inplace=True))
        elif activation == "linear":
            pass
        else:
            print("activate error !!! {} {} {}".format(sys._getframe().f_code.co_
↪filename,
                                                       sys._getframe().f_code.co_name,␣
↪sys._getframe().f_lineno))

    def forward(self, x):
        for l in self.conv:
            x = l(x)
        return x


class ResBlock(nn.Module):
    """
    Sequential residual blocks each of which consists of \
    two convolution layers.
    Args:
        ch (int): number of input and output channels.
        nblocks (int): number of residual blocks.
        shortcut (bool): if True, residual tensor addition is enabled.
    """

    def __init__(self, ch, nblocks=1, shortcut=True):
        super().__init__()
        self.shortcut = shortcut
        self.module_list = nn.ModuleList()
        for i in range(nblocks):
            resblock_one = nn.ModuleList()
            resblock_one.append(Conv_Bn_Activation(ch, ch, 1, 1, 'mish'))
            resblock_one.append(Conv_Bn_Activation(ch, ch, 3, 1, 'mish'))
            self.module_list.append(resblock_one)

    def forward(self, x):
        for module in self.module_list:
            h = x
            for res in module:
                h = res(h)
```

(continues on next page)

(continued from previous page)

```
            x = x + h if self.shortcut else h
        return x


class DownSample1(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = Conv_Bn_Activation(3, 32, 3, 1, 'mish')

        self.conv2 = Conv_Bn_Activation(32, 64, 3, 2, 'mish')
        self.conv3 = Conv_Bn_Activation(64, 64, 1, 1, 'mish')
        # [route]
        # layers = -2
        self.conv4 = Conv_Bn_Activation(64, 64, 1, 1, 'mish')

        self.conv5 = Conv_Bn_Activation(64, 32, 1, 1, 'mish')
        self.conv6 = Conv_Bn_Activation(32, 64, 3, 1, 'mish')
        # [shortcut]
        # from=-3
        # activation = linear

        self.conv7 = Conv_Bn_Activation(64, 64, 1, 1, 'mish')
        # [route]
        # layers = -1, -7
        self.conv8 = Conv_Bn_Activation(128, 64, 1, 1, 'mish')

    def forward(self, input):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x2)
        # route -2
        x4 = self.conv4(x2)
        x5 = self.conv5(x4)
        x6 = self.conv6(x5)
        # shortcut -3
        x6 = x6 + x4

        x7 = self.conv7(x6)
        # [route]
        # layers = -1, -7
        x7 = torch.cat([x7, x3], dim=1)
        x8 = self.conv8(x7)
        return x8


class DownSample2(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = Conv_Bn_Activation(64, 128, 3, 2, 'mish')
        self.conv2 = Conv_Bn_Activation(128, 64, 1, 1, 'mish')
        # r -2
        self.conv3 = Conv_Bn_Activation(128, 64, 1, 1, 'mish')
```

(continues on next page)

(continued from previous page)

```
        self.resblock = ResBlock(ch=64, nblocks=2)

        # s -3
        self.conv4 = Conv_Bn_Activation(64, 64, 1, 1, 'mish')
        # r -1 -10
        self.conv5 = Conv_Bn_Activation(128, 128, 1, 1, 'mish')

    def forward(self, input):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x1)

        r = self.resblock(x3)
        x4 = self.conv4(r)

        x4 = torch.cat([x4, x2], dim=1)
        x5 = self.conv5(x4)
        return x5


class DownSample3(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = Conv_Bn_Activation(128, 256, 3, 2, 'mish')
        self.conv2 = Conv_Bn_Activation(256, 128, 1, 1, 'mish')
        self.conv3 = Conv_Bn_Activation(256, 128, 1, 1, 'mish')

        self.resblock = ResBlock(ch=128, nblocks=8)
        self.conv4 = Conv_Bn_Activation(128, 128, 1, 1, 'mish')
        self.conv5 = Conv_Bn_Activation(256, 256, 1, 1, 'mish')

    def forward(self, input):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x1)

        r = self.resblock(x3)
        x4 = self.conv4(r)

        x4 = torch.cat([x4, x2], dim=1)
        x5 = self.conv5(x4)
        return x5


class DownSample4(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = Conv_Bn_Activation(256, 512, 3, 2, 'mish')
        self.conv2 = Conv_Bn_Activation(512, 256, 1, 1, 'mish')
        self.conv3 = Conv_Bn_Activation(512, 256, 1, 1, 'mish')
```

(continues on next page)

(continued from previous page)

```
        self.resblock = ResBlock(ch=256, nblocks=8)
        self.conv4 = Conv_Bn_Activation(256, 256, 1, 1, 'mish')
        self.conv5 = Conv_Bn_Activation(512, 512, 1, 1, 'mish')

    def forward(self, input):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x1)

        r = self.resblock(x3)
        x4 = self.conv4(r)

        x4 = torch.cat([x4, x2], dim=1)
        x5 = self.conv5(x4)
        return x5


class DownSample5(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = Conv_Bn_Activation(512, 1024, 3, 2, 'mish')
        self.conv2 = Conv_Bn_Activation(1024, 512, 1, 1, 'mish')
        self.conv3 = Conv_Bn_Activation(1024, 512, 1, 1, 'mish')

        self.resblock = ResBlock(ch=512, nblocks=4)
        self.conv4 = Conv_Bn_Activation(512, 512, 1, 1, 'mish')
        self.conv5 = Conv_Bn_Activation(1024, 1024, 1, 1, 'mish')

    def forward(self, input):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x1)

        r = self.resblock(x3)
        x4 = self.conv4(r)

        x4 = torch.cat([x4, x2], dim=1)
        x5 = self.conv5(x4)
        return x5


class Neck(nn.Module):
    def __init__(self, inference=False):
        super().__init__()
        self.inference = inference

        self.conv1 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        self.conv2 = Conv_Bn_Activation(512, 1024, 3, 1, 'leaky')
        self.conv3 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        # SPP
        self.maxpool1 = nn.MaxPool2d(kernel_size=5, stride=1, padding=5 // 2)
        self.maxpool2 = nn.MaxPool2d(kernel_size=9, stride=1, padding=9 // 2)
```

(continues on next page)

(continued from previous page)

```
        self.maxpool3 = nn.MaxPool2d(kernel_size=13, stride=1, padding=13 // 2)

        # R -1 -3 -5 -6
        # SPP
        self.conv4 = Conv_Bn_Activation(2048, 512, 1, 1, 'leaky')
        self.conv5 = Conv_Bn_Activation(512, 1024, 3, 1, 'leaky')
        self.conv6 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        self.conv7 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        # UP
        self.upsample1 = Upsample()
        # R 85
        self.conv8 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        # R -1 -3
        self.conv9 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv10 = Conv_Bn_Activation(256, 512, 3, 1, 'leaky')
        self.conv11 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv12 = Conv_Bn_Activation(256, 512, 3, 1, 'leaky')
        self.conv13 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv14 = Conv_Bn_Activation(256, 128, 1, 1, 'leaky')
        # UP
        self.upsample2 = Upsample()
        # R 54
        self.conv15 = Conv_Bn_Activation(256, 128, 1, 1, 'leaky')
        # R -1 -3
        self.conv16 = Conv_Bn_Activation(256, 128, 1, 1, 'leaky')
        self.conv17 = Conv_Bn_Activation(128, 256, 3, 1, 'leaky')
        self.conv18 = Conv_Bn_Activation(256, 128, 1, 1, 'leaky')
        self.conv19 = Conv_Bn_Activation(128, 256, 3, 1, 'leaky')
        self.conv20 = Conv_Bn_Activation(256, 128, 1, 1, 'leaky')

    def forward(self, input, downsample4, downsample3, inference=False):
        x1 = self.conv1(input)
        x2 = self.conv2(x1)
        x3 = self.conv3(x2)
        # SPP
        m1 = self.maxpool1(x3)
        m2 = self.maxpool2(x3)
        m3 = self.maxpool3(x3)
        spp = torch.cat([m3, m2, m1, x3], dim=1)
        # SPP end
        x4 = self.conv4(spp)
        x5 = self.conv5(x4)
        x6 = self.conv6(x5)
        x7 = self.conv7(x6)
        # UP
        up = self.upsample1(x7, downsample4.size(), self.inference)
        # R 85
        x8 = self.conv8(downsample4)
        # R -1 -3
        x8 = torch.cat([x8, up], dim=1)

        x9 = self.conv9(x8)
```

(continues on next page)

(continued from previous page)

```
        x10 = self.conv10(x9)
        x11 = self.conv11(x10)
        x12 = self.conv12(x11)
        x13 = self.conv13(x12)
        x14 = self.conv14(x13)

        # UP
        up = self.upsample2(x14, downsample3.size(), self.inference)
        # R 54
        x15 = self.conv15(downsample3)
        # R -1 -3
        x15 = torch.cat([x15, up], dim=1)

        x16 = self.conv16(x15)
        x17 = self.conv17(x16)
        x18 = self.conv18(x17)
        x19 = self.conv19(x18)
        x20 = self.conv20(x19)
        return x20, x13, x6


class Yolov4Head(nn.Module):
    def __init__(self, output_ch, n_classes, inference=False):
        super().__init__()
        self.inference = inference

        self.conv1 = Conv_Bn_Activation(128, 256, 3, 1, 'leaky')
        self.conv2 = Conv_Bn_Activation(256, output_ch, 1, 1, 'linear', bn=False,␣
↪bias=True)

        self.yolo1 = YoloLayer(
                                anchor_mask=[0, 1, 2], num_classes=n_classes,
                                anchors=[12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146,
↪ 142, 110, 192, 243, 459, 401],
                                num_anchors=9, stride=8)

        # R -4
        self.conv3 = Conv_Bn_Activation(128, 256, 3, 2, 'leaky')

        # R -1 -16
        self.conv4 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv5 = Conv_Bn_Activation(256, 512, 3, 1, 'leaky')
        self.conv6 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv7 = Conv_Bn_Activation(256, 512, 3, 1, 'leaky')
        self.conv8 = Conv_Bn_Activation(512, 256, 1, 1, 'leaky')
        self.conv9 = Conv_Bn_Activation(256, 512, 3, 1, 'leaky')
        self.conv10 = Conv_Bn_Activation(512, output_ch, 1, 1, 'linear', bn=False,␣
↪bias=True)

        self.yolo2 = YoloLayer(
                                anchor_mask=[3, 4, 5], num_classes=n_classes,
                                anchors=[12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146,
↪ 142, 110, 192, 243, 459, 401],
```

(continues on next page)

(continued from previous page)

```
                                num_anchors=9, stride=16)

        # R -4
        self.conv11 = Conv_Bn_Activation(256, 512, 3, 2, 'leaky')

        # R -1 -37
        self.conv12 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        self.conv13 = Conv_Bn_Activation(512, 1024, 3, 1, 'leaky')
        self.conv14 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        self.conv15 = Conv_Bn_Activation(512, 1024, 3, 1, 'leaky')
        self.conv16 = Conv_Bn_Activation(1024, 512, 1, 1, 'leaky')
        self.conv17 = Conv_Bn_Activation(512, 1024, 3, 1, 'leaky')
        self.conv18 = Conv_Bn_Activation(1024, output_ch, 1, 1, 'linear', bn=False,␣
↪bias=True)

        self.yolo3 = YoloLayer(
                                anchor_mask=[6, 7, 8], num_classes=n_classes,
                                anchors=[12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146,
↪ 142, 110, 192, 243, 459, 401],
                                num_anchors=9, stride=32)

    def forward(self, input1, input2, input3):
        x1 = self.conv1(input1)
        x2 = self.conv2(x1)

        x3 = self.conv3(input1)
        # R -1 -16
        x3 = torch.cat([x3, input2], dim=1)
        x4 = self.conv4(x3)
        x5 = self.conv5(x4)
        x6 = self.conv6(x5)
        x7 = self.conv7(x6)
        x8 = self.conv8(x7)
        x9 = self.conv9(x8)
        x10 = self.conv10(x9)

        # R -4
        x11 = self.conv11(x8)
        # R -1 -37
        x11 = torch.cat([x11, input3], dim=1)

        x12 = self.conv12(x11)
        x13 = self.conv13(x12)
        x14 = self.conv14(x13)
        x15 = self.conv15(x14)
        x16 = self.conv16(x15)
        x17 = self.conv17(x16)
        x18 = self.conv18(x17)

        if self.inference:
            y1 = self.yolo1(x2)
            y2 = self.yolo2(x10)
```

(continues on next page)

(continued from previous page)

```
            y3 = self.yolo3(x18)

            return get_region_boxes([y1, y2, y3])

        else:
            return [x2, x10, x18]


class Yolov4(nn.Module):
    def __init__(self, yolov4conv137weight=None, n_classes=80, inference=False):
        super().__init__()

        output_ch = (4 + 1 + n_classes) * 3

        # backbone
        self.down1 = DownSample1()
        self.down2 = DownSample2()
        self.down3 = DownSample3()
        self.down4 = DownSample4()
        self.down5 = DownSample5()
        # neck
        self.neek = Neck(inference)
        # yolov4conv137
        if yolov4conv137weight:
            _model = nn.Sequential(self.down1, self.down2, self.down3, self.down4, self.
↪down5, self.neek)
            pretrained_dict = torch.load(yolov4conv137weight)

            model_dict = _model.state_dict()
            # 1. filter out unnecessary keys
            pretrained_dict = {k1: v for (k, v), k1 in zip(pretrained_dict.items(),␣
↪model_dict)}
            # 2. overwrite entries in the existing state dict
            model_dict.update(pretrained_dict)
            _model.load_state_dict(model_dict)

        # head
        self.head = Yolov4Head(output_ch, n_classes, inference)


    def forward(self, input):
        d1 = self.down1(input)
        d2 = self.down2(d1)
        d3 = self.down3(d2)
        d4 = self.down4(d3)
        d5 = self.down5(d4)

        x20, x13, x6 = self.neek(d5, d4, d3)

        output = self.head(x20, x13, x6)
        return output
```

(continues on next page)

(continued from previous page)

```
def yolo_forward_dynamic(output, conf_thresh, num_classes, anchors, num_anchors, scale_x_
↪y, only_objectness=1,
                             validation=False):
    # Output would be invalid if it does not satisfy this assert
    # assert (output.size(1) == (5 + num_classes) * num_anchors)

    # print(output.size())

    # Slice the second dimension (channel) of output into:
    # [ 2, 2, 1, num_classes, 2, 2, 1, num_classes, 2, 2, 1, num_classes ]
    # And then into
    # bxy = [ 6 ] bwh = [ 6 ] det_conf = [ 3 ] cls_conf = [ num_classes * 3 ]
    # batch = output.size(0)
    # H = output.size(2)
    # W = output.size(3)

    bxy_list = []
    bwh_list = []
    det_confs_list = []
    cls_confs_list = []

    for i in range(num_anchors):
        begin = i * (5 + num_classes)
        end = (i + 1) * (5 + num_classes)

        bxy_list.append(output[:, begin : begin + 2])
        bwh_list.append(output[:, begin + 2 : begin + 4])
        det_confs_list.append(output[:, begin + 4 : begin + 5])
        cls_confs_list.append(output[:, begin + 5 : end])

    # Shape: [batch, num_anchors * 2, H, W]
    bxy = torch.cat(bxy_list, dim=1)
    # Shape: [batch, num_anchors * 2, H, W]
    bwh = torch.cat(bwh_list, dim=1)

    # Shape: [batch, num_anchors, H, W]
    det_confs = torch.cat(det_confs_list, dim=1)
    # Shape: [batch, num_anchors * H * W]
    det_confs = det_confs.view(output.size(0), num_anchors * output.size(2) * output.
↪size(3))

    # Shape: [batch, num_anchors * num_classes, H, W]
    cls_confs = torch.cat(cls_confs_list, dim=1)
    # Shape: [batch, num_anchors, num_classes, H * W]
    cls_confs = cls_confs.view(output.size(0), num_anchors, num_classes, output.size(2)␣
↪* output.size(3))
    # Shape: [batch, num_anchors, num_classes, H * W] --> [batch, num_anchors * H * W,␣
↪num_classes]
    cls_confs = cls_confs.permute(0, 1, 3, 2).reshape(output.size(0), num_anchors *␣
↪output.size(2) * output.size(3), num_classes)
```

(continues on next page)

(continued from previous page)

```
    # Apply sigmoid(), exp() and softmax() to slices
    #
    bxy = torch.sigmoid(bxy) * scale_x_y - 0.5 * (scale_x_y - 1)
    bwh = torch.exp(bwh)
    det_confs = torch.sigmoid(det_confs)
    cls_confs = torch.sigmoid(cls_confs)

    # Prepare C-x, C-y, P-w, P-h (None of them are torch related)
    grid_x = np.expand_dims(np.expand_dims(np.expand_dims(np.linspace(0, output.size(3) -
↪ 1, output.size(3)), axis=0).repeat(output.size(2), 0), axis=0), axis=0)
    grid_y = np.expand_dims(np.expand_dims(np.expand_dims(np.linspace(0, output.size(2) -
↪ 1, output.size(2)), axis=1).repeat(output.size(3), 1), axis=0), axis=0)
    # grid_x = torch.linspace(0, W - 1, W).reshape(1, 1, 1, W).repeat(1, 1, H, 1)
    # grid_y = torch.linspace(0, H - 1, H).reshape(1, 1, H, 1).repeat(1, 1, 1, W)

    anchor_w = []
    anchor_h = []
    for i in range(num_anchors):
        anchor_w.append(anchors[i * 2])
        anchor_h.append(anchors[i * 2 + 1])

    device = None
    cuda_check = output.is_cuda
    if cuda_check:
        device = output.get_device()

    bx_list = []
    by_list = []
    bw_list = []
    bh_list = []

    # Apply C-x, C-y, P-w, P-h
    for i in range(num_anchors):
        ii = i * 2
        # Shape: [batch, 1, H, W]
        bx = bxy[:, ii : ii + 1] + torch.tensor(grid_x, device=device, dtype=torch.
↪float32) # grid_x.to(device=device, dtype=torch.float32)
        # Shape: [batch, 1, H, W]
        by = bxy[:, ii + 1 : ii + 2] + torch.tensor(grid_y, device=device, dtype=torch.
↪float32) # grid_y.to(device=device, dtype=torch.float32)
        # Shape: [batch, 1, H, W]
        bw = bwh[:, ii : ii + 1] * anchor_w[i]
        # Shape: [batch, 1, H, W]
        bh = bwh[:, ii + 1 : ii + 2] * anchor_h[i]

        bx_list.append(bx)
        by_list.append(by)
        bw_list.append(bw)
        bh_list.append(bh)


    ########################################
```

(continues on next page)

(continued from previous page)

```
    #   Figure out bboxes from slices     #
    ########################################

    # Shape: [batch, num_anchors, H, W]
    bx = torch.cat(bx_list, dim=1)
    # Shape: [batch, num_anchors, H, W]
    by = torch.cat(by_list, dim=1)
    # Shape: [batch, num_anchors, H, W]
    bw = torch.cat(bw_list, dim=1)
    # Shape: [batch, num_anchors, H, W]
    bh = torch.cat(bh_list, dim=1)

    # Shape: [batch, 2 * num_anchors, H, W]
    bx_bw = torch.cat((bx, bw), dim=1)
    # Shape: [batch, 2 * num_anchors, H, W]
    by_bh = torch.cat((by, bh), dim=1)

    # normalize coordinates to [0, 1]
    bx_bw /= output.size(3)
    by_bh /= output.size(2)

    # Shape: [batch, num_anchors * H * W, 1]
    bx = bx_bw[:, :num_anchors].view(output.size(0), num_anchors * output.size(2) *␣
↪output.size(3), 1)
    by = by_bh[:, :num_anchors].view(output.size(0), num_anchors * output.size(2) *␣
↪output.size(3), 1)
    bw = bx_bw[:, num_anchors:].view(output.size(0), num_anchors * output.size(2) *␣
↪output.size(3), 1)
    bh = by_bh[:, num_anchors:].view(output.size(0), num_anchors * output.size(2) *␣
↪output.size(3), 1)

    bx1 = bx - bw * 0.5
    by1 = by - bh * 0.5
    bx2 = bx1 + bw
    by2 = by1 + bh

    # Shape: [batch, num_anchors * h * w, 4] -> [batch, num_anchors * h * w, 1, 4]
    boxes = torch.cat((bx1, by1, bx2, by2), dim=2).view(output.size(0), num_anchors *␣
↪output.size(2) * output.size(3), 1, 4)
    # boxes = boxes.repeat(1, 1, num_classes, 1)

    # boxes:     [batch, num_anchors * H * W, 1, 4]
    # cls_confs: [batch, num_anchors * H * W, num_classes]
    # det_confs: [batch, num_anchors * H * W]

    det_confs = det_confs.view(output.size(0), num_anchors * output.size(2) * output.
↪size(3), 1)
    confs = cls_confs * det_confs

    # boxes: [batch, num_anchors * H * W, 1, 4]
    # confs: [batch, num_anchors * H * W, num_classes]
```

(continues on next page)

(continued from previous page)

```
    return  boxes, confs

class YoloLayer(nn.Module):
    """
    Yolo layer
    model_out: while inference,is post-processing inside or outside the model
        true:outside
    """
    def __init__(self, anchor_mask=[], num_classes=0, anchors=[], num_anchors=1,␣
↪stride=32, model_out=False):
        super(YoloLayer, self).__init__()
        self.anchor_mask = anchor_mask
        self.num_classes = num_classes
        self.anchors = anchors
        self.num_anchors = num_anchors
        self.anchor_step = len(anchors) // num_anchors
        self.coord_scale = 1
        self.noobject_scale = 1
        self.object_scale = 5
        self.class_scale = 1
        self.thresh = 0.6
        self.stride = stride
        self.seen = 0
        self.scale_x_y = 1

        self.model_out = model_out

    def forward(self, output, target=None):
        if self.training:
            return output
        masked_anchors = []
        for m in self.anchor_mask:
            masked_anchors += self.anchors[m * self.anchor_step:(m + 1) * self.anchor_
↪step]
        masked_anchors = [anchor / self.stride for anchor in masked_anchors]

        return yolo_forward_dynamic(output, self.thresh, self.num_classes, masked_
↪anchors, len(self.anchor_mask),scale_x_y=self.scale_x_y)


def get_region_boxes(boxes_and_confs):

    # print('Getting boxes from boxes and confs ...')

    boxes_list = []
    confs_list = []

    for item in boxes_and_confs:
        boxes_list.append(item[0])
        confs_list.append(item[1])

    # boxes: [batch, num1 + num2 + num3, 1, 4]
```

(continues on next page)

(continued from previous page)

```
    # confs: [batch, num1 + num2 + num3, num_classes]
    boxes = torch.cat(boxes_list, dim=1)
    confs = torch.cat(confs_list, dim=1)

    return boxes, confs
```

8.22.4 Part 2: Download the COCO 2017 evaluation dataset and define the data loader function

Download dataset

[ ]:
```
!curl -LO http://images.cocodataset.org/zips/val2017.zip
!curl -LO http://images.cocodataset.org/annotations/annotations_trainval2017.zip
!unzip -q val2017.zip
!unzip annotations_trainval2017.zip
```

[ ]:
```
!ls
```

Define data loader

[ ]:
```
import os
import json
import time
import torchvision
import torchvision.transforms as transforms
import torchvision.datasets as dset
from pycocotools.coco import COCO


def get_image_filenames(root=os.getcwd()):
    """
    Generate paths to the coco dataset image files.

    Args:
        root (str): The root folder contains.

    Yields:
        filename (str): The path to an image file.
    """
    image_path = os.path.join(root, 'val2017')
    for root, dirs, files in os.walk(image_path):
        for filename in files:
            yield os.path.join(image_path, filename)


def get_coco_dataloader(coco2017_root, transform, subset_indices=None):
    """
    Create the dataset loader and ground truth coco dataset.
```

(continues on next page)

(continued from previous page)

```
    Arguments:
        coco2017_root (str): The root directory to load the data/labels from.
        transform (torchvision.Transform): A transform to apply to the images.
        subset_indices (list): Indices used to create a subset of the dataset.

    Returns:
        loader (iterable): Produces transformed images and labels.
        cocoGt (pycocotools.coco.COCO): Contains the ground truth in coco
            format.
        label_info (dict): A mapping from label id to the human-readable name.
    """

    # Create the dataset
    coco2017_img_path = os.path.join(coco2017_root, 'val2017')
    coco2017_ann_path = os.path.join(
        coco2017_root, 'annotations/instances_val2017.json')

    # check the number of images in val2017 - Should be 5000
    num_files = len(list(get_image_filenames(coco2017_root)))
    print('\nNumber of images in val2017 = {}\n'.format(num_files))

    # load annotations to decode classification results
    with open(coco2017_ann_path) as f:
        annotate_json = json.load(f)
    label_info = {label["id"]: label["name"]
                  for label in annotate_json['categories']}

    # initialize COCO ground truth dataset
    cocoGt = COCO(coco2017_ann_path)

    # create the dataset using torchvision's coco detection dataset
    coco_val_data = dset.CocoDetection(
        root=coco2017_img_path,
        annFile=coco2017_ann_path,
        transform=transform
    )

    if subset_indices is not None:
        # Create a smaller subset of the data for testing - e.g. to pinpoint error at␣
↪image 516
        coco_val_data = torch.utils.data.Subset(coco_val_data, subset_indices)

    # create the dataloader using torch dataloader
    loader = torch.utils.data.DataLoader(coco_val_data, batch_size=1, shuffle=False)

    return loader, cocoGt, label_info
```

Load dataset

Here 2 dataset loaders are created and the resulting data is displayed - `orig_coco_val_data_loader`: Contains the original unmodified image - `coco_val_data_loader`: Contains images of a standardized size of 608x608 pixels

[ ]:
```
coco2017_root = './'
orig_coco_val_data_loader, *_ = get_coco_dataloader(coco2017_root, transforms.ToTensor())
transform = transforms.Compose([transforms.Resize([608, 608]), transforms.ToTensor()])
coco_val_data_loader, cocoGt, label_info = get_coco_dataloader(coco2017_root, transform)
image_orig, _ = next(iter(orig_coco_val_data_loader))
print(image_orig.shape)
image, image_info = next(iter(coco_val_data_loader))
image_id = image_info[0]["image_id"].item()
print(image.shape)
```

Define some helper functions for deployment (inference)

[ ]:
```
def postprocess(boxes, scores, score_threshold=0.05, iou_threshold=0.5):
    """
    Classifies and filters bounding boxes from Yolo V4 output.

    Performs classification, filtering, and non-maximum suppression to remove
    boxes that are irrelevant. The result is the filtered set of boxes, the
    associated label confidence score, and the predicted label.

    See: https://pytorch.org/docs/stable/torchvision/ops.html#torchvision.ops.nms

    Args:
        boxes (torch.Tensor): The Yolo V4 bounding boxes.
        scores (torch.Tensor): The categories scores for each box.
        score_threshold (float): Ignore boxes with scores below threshold.
        iou_threshold (float): Discards boxes with intersection above threshold.

    Returns:
        boxes (torch.Tensor): The filtered Yolo V4 bounding boxes.
        scores (torch.Tensor): The label score for each box.
        labels (torch.Tensor): The label for each box.
    """

    # shape: [n_batch, n_boxes, 1, 4] => [n_boxes, 4]  # Assumes n_batch size is 1
    boxes = boxes.squeeze()

    # shape: [n_batch, n_boxes, 80] => [n_boxes, 80]  # Assumes n_batch size is 1
    scores = scores.squeeze()

    # Classify each box according to the maximum category score
    score, column = torch.max(scores, dim=1)

    # Filter out rows for scores which are below threshold
    mask = score > score_threshold

    # Filter model output data
```

(continues on next page)

(continued from previous page)

```
    boxes = boxes[mask]
    score = score[mask]
    idxs = column[mask]

    # Perform non-max suppression on all categories at once. shape: [n_keep,]
    keep = torchvision.ops.batched_nms(
        boxes=boxes,
        scores=score,
        idxs=idxs,
        iou_threshold=iou_threshold,
    )

    # The image category id associated with each column
    categories = torch.tensor([
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 31,
        32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
        44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
        57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 70, 72,
        73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 85,
        86, 87, 88, 89, 90
    ])

    boxes = boxes[keep]       # shape: [n_keep, 4]
    score = score[keep]       # shape: [n_keep,]
    idxs = idxs[keep]
    label = categories[idxs]  # shape: [n_keep,]

    return boxes, score, label


def get_results_as_dict(boxes, scores, labels, image_orig):
    """
    Transforms post-processed output into dictionary output.

    This translates the model coordinate bounding boxes (x1, y1, x2, y2)
    into a rectangular description (x, y, width, height) scaled to the
    original image size.

    Args:
        boxes (torch.Tensor): The Yolo V4 bounding boxes.
        scores (torch.Tensor): The label score for each box.
        labels (torch.Tensor): The label for each box.
        image_orig (torch.Tensor): The image to scale the bounding boxes to.

    Returns:
        output (dict): The dictionary of rectangle bounding boxes.
    """
    h_size, w_size = image_orig.shape[-2:]

    x1 = boxes[:, 0] * w_size
    y1 = boxes[:, 1] * h_size
```

(continues on next page)

(continued from previous page)

```
    x2 = boxes[:, 2] * w_size
    y2 = boxes[:, 3] * h_size

    width = x2 - x1
    height = y2 - y1

    boxes = torch.stack([x1, y1, width, height]).T
    return {
        'boxes': boxes.detach().numpy(),
        'labels': labels.detach().numpy(),
        'scores': scores.detach().numpy(),
    }


def prepare_for_coco_detection(predictions):
    """
    Convert dictionary model predictions into an expected COCO dataset format.

    Args:
        predictions (dict): The list of box coordinates, scores, and labels.

    Returns:
        output (list[dict]): The list of bounding boxes.
    """
    coco_results = []
    for original_id, prediction in predictions.items():
        if len(prediction) == 0:
            continue

        boxes = prediction["boxes"].tolist()
        scores = prediction["scores"].tolist()
        labels = prediction["labels"].tolist()

        coco_results.extend(
            [
                {
                    "image_id": original_id,
                    "category_id": labels[k],
                    "bbox": box,
                    "score": scores[k],
                }
                for k, box in enumerate(boxes)
            ]
        )
    return coco_results
```

Download pretrained checkpoint

[ ]:
```
import requests

def download_file_from_google_drive(id, destination):
    response = requests.post('https://drive.google.com/uc?id='+id+'&confirm=t')
    save_response_content(response, destination)

def save_response_content(response, destination):
    CHUNK_SIZE = 32768
    with open(destination, "wb") as f:
        for chunk in response.iter_content(CHUNK_SIZE):
            if chunk:  # filter out keep-alive new chunks
                f.write(chunk)
```

[ ]:
```
download_file_from_google_drive('1wv_LiFeCRYwtpkqREPeI13-gPELBDwuJ', './yolo_v4.pth')
```

8.22.5 Part 3: Build, Compile, and Save Neuron-Optimized YOLO v4 TorchScript

Construct model and load pretrained checkpoint

[ ]:
```
model = Yolov4(yolov4conv137weight=None, n_classes=80, inference=True)
weightfile = "./yolo_v4.pth"
pretrained_dict = torch.load(weightfile, map_location=torch.device('cpu'))
model.load_state_dict(pretrained_dict)
model.eval()
```

Execute inference for a single image and display output

[ ]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

image_orig, _ = next(iter(orig_coco_val_data_loader))
image, _ = next(iter(coco_val_data_loader))
boxes, scores = model(image)
boxes, scores, labels = postprocess(boxes, scores)
result_dict = get_results_as_dict(boxes, scores, labels, image_orig)

fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(image_orig.numpy().squeeze(0).transpose(1, 2, 0))
for xywh, _ in zip(result_dict['boxes'], result_dict['labels']):
    x, y, w, h = xywh
    rect = patches.Rectangle((x, y), w, h, linewidth=1, edgecolor='g', facecolor='none')
    ax.add_patch(rect)
plt.show()
```

Run compilation with manually specified device placement

First, inspect the model without running compilation by adding the `skip_compiler=True` argument to the `torch.neuron.trace` call.

```
[ ]: model_neuron_for_inspection = torch.neuron.trace(model, image, skip_compiler=True)
print(model_neuron_for_inspection)
```

Inspecting the model, we discover that there are many `aten::slice` operations in some submodules called `YoloLayer`. Although these operations are supported by the neuron-cc compiler, they are not going to run efficiently on the Inferentia hardware. To work it around, we recommend to manually place these operators on CPU.

To manually place `YoloLayer` on CPU, we may make use of the `subgraph_builder_function` argument in `torch.neuron.trace`. It is a callback function that returns `True` or `False` based on information available in `node`. The typical use is a condition based on either `node.name` or `node.type_string`.

```
[ ]: def subgraph_builder_function(node):
    return 'YoloLayer' not in node.name

model_neuron = torch.neuron.trace(model, image, subgraph_builder_function=subgraph_
↪builder_function)
model_neuron.save('yolo_v4_neuron.pt')
```

Compilation is now finished and the compiled model has been saved to a local file called 'yolo_v4_neuron.pt'. Saving is important due to the slow compilation process.

8.22.6 Part 4: Evaluate Accuracy on the COCO 2017 Dataset

Load compiled model and run inference

To validate accuracy of the compiled model, lets run inference on the COCO 2017 validation dataset. We start by defining a helper function `run_inference`.

```
[ ]: def run_inference(dataloader, dataloader_orig, model, convert=True, modelName=''):
    """
    Run Yolo V4 inference on the COCO dataset.

    Args:
        dataloader (iterable): Data loader of input processed images and labels.
        dataloader_orig (iterable): Data loader with original images.
        model (torch.nn.Module): The torch model to run inference against.
        convert (bool): Set to False when using a vanilla torchvision model that
            does not need to be transformed into coco format.

    Returns:
        imgIds (list): The list of images with predictions.
        cocoDt (pycocotools.coco.COCO): Contains the predictions from the model
            in coco format.
    """
    print('\n================ Starting Inference on {} Images using {} model␣
↪================\n'.format(
        len(dataloader), modelName))
```

(continues on next page)

(continued from previous page)

```
    modelName = str(modelName).replace(" ", "_")

    # convert predicition to cocoDt
    # code from def evaluate in https://github.com/pytorch/vision/blob/master/references/
↪detection/engine.py
    imgIds = []
    results = []
    skippedImages = []

    # time inference
    inference_time = 0.0
    for idx, ((image, targets), (image_orig, _)) in enumerate(zip(dataloader, dataloader_
↪orig)):
        # if target is empty, skip the image because it breaks the scripted model
        if not targets:
            skippedImages.append(idx)
            continue

        # get the predictions
        start_time = time.time()
        boxes, scores = model(image)
        delta = time.time() - start_time
        inference_time += delta
        boxes, scores, labels = postprocess(boxes, scores)
        outputs = get_results_as_dict(boxes, scores, labels, image_orig)

        res = {target["image_id"].item(): output for target,
               output in zip(targets, [outputs])}

        # add the image id to imgIds
        image_id = targets[0]["image_id"].item()
        imgIds.append(image_id)

        # convert the predicition into cocoDt results
        pred = prepare_for_coco_detection(res)
        results.extend(pred)

    print('\n==================== Performance Measurement ====================')
    print('Finished inference on {} images in {:.2f} seconds'.format(
        len(dataloader), inference_time))
    print('=================================================================\n')

    # create bbox detections file
    # following code in https://github.com/aws/aws-neuron-sdk/blob/master/src/examples/
↪tensorflow/yolo_v4_demo/evaluate.ipynb
    resultsfile = modelName + '_bbox_detections.json'
    print('Generating json file...')
    with open(resultsfile, 'w') as f:
        json.dump(results, f)

    # return COCO api object with loadRes
    cocoDt = cocoGt.loadRes(resultsfile)
```

(continues on next page)

(continued from previous page)

```
    return imgIds, cocoDt
```

The next step is to simply load the compiled model from disk and then run inference.

```
[ ]: model_neuron = torch.jit.load('yolo_v4_neuron.pt')
```

```
[ ]: imgIds, cocoDt = run_inference(coco_val_data_loader, orig_coco_val_data_loader, model_
     ↪neuron)
```

We then use the standard `pycocotools` routines to generate a report of bounding box precision/recall.

```
[ ]: from pycocotools.cocoeval import COCOeval

     cocoEval = COCOeval(cocoGt, cocoDt, 'bbox')
     cocoEval.params.imgIds = imgIds
     cocoEval.evaluate()
     cocoEval.accumulate()
     cocoEval.summarize()
```

For reference, we may perform the same evaluation on the CPU model.

```
[ ]: imgIdsRef, cocoDtRef = run_inference(coco_val_data_loader, orig_coco_val_data_loader,␣
     ↪model)
```

```
[ ]: cocoEval = COCOeval(cocoGt, cocoDtRef, 'bbox')
     cocoEval.params.imgIds = imgIdsRef
     cocoEval.evaluate()
     cocoEval.accumulate()
     cocoEval.summarize()
```

8.22.7 Part 5: Benchmark COCO Dataset Performance of the Neuron-Optimized TorchScript

The following code snippet sets up data parallel on 16 NeuronCores and runs saturated multi-threaded inference on the Inferentia accelerator. Note that the number of cores (`n_cores`) should be set to the number of available NeuronCores on the current instance.

```
[ ]: import torch
     import torch.neuron
     import torchvision
     import torchvision.transforms as transforms
     import torchvision.datasets as dset
     import multiprocessing as mp
     from concurrent.futures import ThreadPoolExecutor
     import PIL
     import os
     import time

     n_threads = 16
```

(continues on next page)

(continued from previous page)

```
def get_image_filenames(root=os.getcwd()):
    """
    Generate paths to the coco dataset image files.

    Args:
        root (str): The root folder contains.

    Yields:
        filename (str): The path to an image file.
    """
    image_path = os.path.join(root, 'val2017')
    for root, dirs, files in os.walk(image_path):
        for filename in files:
            yield os.path.join(image_path, filename)

def preprocess(path):
    """
    Load an image and convert to the expected Yolo V4 tensor format.

    Args:
        path (str): The image file to load from disk.

    Returns:
        result (torch.Tensor): The image for prediction. Shape: [1, 3, 608, 608]
    """
    image = PIL.Image.open(path).convert('RGB')
    resized = torchvision.transforms.functional.resize(image, [608, 608])
    tensor = torchvision.transforms.functional.to_tensor(resized)
    return tensor.unsqueeze(0).to(torch.float32)


def load_model(filename='yolo_v4_neuron.pt'):
    """
    Load and pre-warm the Yolo V4 model.

    Args:
        filename (str): The location to load the model from.

    Returns:
        model (torch.nn.Module): The torch model.
    """

    # Load model from disk
    model = torch.jit.load(filename)

    # Warm up model on neuron by running a single example image
    filename = next(iter(get_image_filenames()))
    image = preprocess(filename)
    model(image)

    return model
```

(continues on next page)

(continued from previous page)

```
def task(model, filename):
    """
    The thread task to perform prediction.

    This does the full end-to-end processing of an image from loading from disk
    all the way to classifying and filtering bounding boxes.

    Args:
        model (torch.nn.Module): The model to run processing with
        filename (str): The image file to load from disk.

    Returns:
        boxes (torch.Tensor): The Yolo V4 bounding boxes.
        scores (torch.Tensor): The label score for each box.
        labels (torch.Tensor): The label for each box.
    """
    image = preprocess(filename)
    begin = time.time()
    boxes, scores = model(image)
    delta = time.time() - begin
    return postprocess(boxes, scores), delta


def benchmark():
    """
    Run a benchmark on the entire COCO dataset against the neuron model.
    """

    # Load a model into each NeuronCore
    models = [load_model() for _ in range(n_cores)]

    # Create input/output lists
    filenames = list(get_image_filenames())
    results = list()
    latency = list()

    # We want to keep track of average completion time per thread
    sum_time = 0.0

    # Submit all tasks and wait for them to finish
    with ThreadPoolExecutor(n_threads) as pool:
        for i, filename in enumerate(filenames):
            result = pool.submit(task, models[i % len(models)], filename)
            results.append(result)
        for result in results:
            results, times = result.result() # Note: Outputs unused for benchmark
            latency.append(times)
            sum_time += times

    print('Duration: ', sum_time / n_threads)
    print('Images Per Second:', len(filenames) / (sum_time / n_threads))
```

(continues on next page)

(continued from previous page)

```
    print("Latency P50: {:.1f}".format(np.percentile(latency[1000:], 50)*1000.0))
    print("Latency P90: {:.1f}".format(np.percentile(latency[1000:], 90)*1000.0))
    print("Latency P95: {:.1f}".format(np.percentile(latency[1000:], 95)*1000.0))
    print("Latency P99: {:.1f}".format(np.percentile(latency[1000:], 99)*1000.0))

benchmark()
```

Natural Language Processing (NLP) Tutorials

- HuggingFace pretrained BERT tutorial *[html]* [notebook]
- Bring your own HuggingFace pretrained BERT container to Sagemaker Tutorial *[html]* [notebook]
- LibTorch C++ tutorial [html]
- HuggingFace MarianMT tutorial *[html]* [notebook]

8.23 Compiling and Deploying HuggingFace Pretrained BERT

8.23.1 Introduction

In this tutorial we will compile and deploy BERT-base version of HuggingFace Transformers BERT for Inferentia. The full list of HuggingFace's pretrained BERT models can be found in the BERT section on this page https://huggingface.co/transformers/pretrained_models.html.

This Jupyter notebook should be run on an instance which is inf1.6xlarge or larger. The compile part of this tutorial requires inf1.6xlarge and not the inference itself. For simplicity we will run this tutorial on inf1.6xlarge but in real life scenario the compilation should be done on a compute instance and the deployment on inf1 instance to save costs.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

Install Dependencies:

This tutorial requires the following pip packages:

- `torch-neuron`
- `neuron-cc[tensorflow]`
- `transformers`

Most of these packages will be installed when configuring your environment using the Neuron PyTorch setup guide. The additional dependencies must be installed here.

```
[ ]: !pip install --upgrade "transformers==4.6.0"
```

8.23.2 Compile the model into an AWS Neuron optimized TorchScript

```
[ ]: import tensorflow  # to workaround a protobuf version conflict issue
import torch
import torch.neuron
from transformers import AutoTokenizer, AutoModelForSequenceClassification, AutoConfig
import transformers
import os
import warnings

# Setting up NeuronCore groups for inf1.6xlarge with 16 cores
num_cores = 16 # This value should be 4 on inf1.xlarge and inf1.2xlarge
os.environ['NEURON_RT_NUM_CORES'] = str(num_cores)

# Build tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased-finetuned-mrpc")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased-finetuned-
↪mrpc", return_dict=False)

# Setup some example inputs
sequence_0 = "The company HuggingFace is based in New York City"
sequence_1 = "Apples are especially bad for your health"
sequence_2 = "HuggingFace's headquarters are situated in Manhattan"

max_length=128
paraphrase = tokenizer.encode_plus(sequence_0, sequence_2, max_length=max_length,␣
↪padding='max_length', truncation=True, return_tensors="pt")
not_paraphrase = tokenizer.encode_plus(sequence_0, sequence_1, max_length=max_length,␣
↪padding='max_length', truncation=True, return_tensors="pt")

# Run the original PyTorch model on compilation exaple
paraphrase_classification_logits = model(**paraphrase)[0]

# Convert example inputs to a format that is compatible with TorchScript tracing
example_inputs_paraphrase = paraphrase['input_ids'], paraphrase['attention_mask'],␣
↪paraphrase['token_type_ids']
example_inputs_not_paraphrase = not_paraphrase['input_ids'], not_paraphrase['attention_
↪mask'], not_paraphrase['token_type_ids']

# Run torch.neuron.trace to generate a TorchScript that is optimized by AWS Neuron
model_neuron = torch.neuron.trace(model, example_inputs_paraphrase)

# Verify the TorchScript works on both example inputs
paraphrase_classification_logits_neuron = model_neuron(*example_inputs_paraphrase)
not_paraphrase_classification_logits_neuron = model_neuron(*example_inputs_not_
↪paraphrase)

# Save the TorchScript for later use
model_neuron.save('bert_neuron.pt')
```

You may inspect `model_neuron.graph` to see which part is running on CPU versus running on the accelerator. All native `aten` operators in the graph will be running on CPU.

```
[ ]: print(model_neuron.graph)
```

8.23.3 Deploy the AWS Neuron optimized TorchScript

To deploy the AWS Neuron optimized TorchScript, you may choose to load the saved TorchScript from disk and skip the slow compilation.

```
[ ]: # Load TorchScript back
model_neuron = torch.jit.load('bert_neuron.pt')
# Verify the TorchScript works on both example inputs
paraphrase_classification_logits_neuron = model_neuron(*example_inputs_paraphrase)
not_paraphrase_classification_logits_neuron = model_neuron(*example_inputs_not_
↪paraphrase)
classes = ['not paraphrase', 'paraphrase']
paraphrase_prediction = paraphrase_classification_logits_neuron[0][0].argmax().item()
not_paraphrase_prediction = not_paraphrase_classification_logits_neuron[0][0].argmax().
↪item()
print('BERT says that "{}" and "{}" are {}'.format(sequence_0, sequence_2,␣
↪classes[paraphrase_prediction]))
print('BERT says that "{}" and "{}" are {}'.format(sequence_0, sequence_1, classes[not_
↪paraphrase_prediction]))
```

Now let's run the model in parallel on four cores

```
[ ]: def get_input_with_padding(batch, batch_size, max_length):
    ## Reformulate the batch into three batch tensors - default batch size batches the␣
↪outer dimension
    encoded = batch['encoded']
    inputs = torch.squeeze(encoded['input_ids'], 1)
    attention = torch.squeeze(encoded['attention_mask'], 1)
    token_type = torch.squeeze(encoded['token_type_ids'], 1)
    quality = list(map(int, batch['quality']))

    if inputs.size()[0] != batch_size:
        print("Input size = {} - padding".format(inputs.size()))
        remainder = batch_size - inputs.size()[0]
        zeros = torch.zeros( [remainder, max_length], dtype=torch.long )
        inputs = torch.cat( [inputs, zeros] )
        attention = torch.cat( [attention, zeros] )
        token_type = torch.cat( [token_type, zeros] )

    assert(inputs.size()[0] == batch_size and inputs.size()[1] == max_length)
    assert(attention.size()[0] == batch_size and attention.size()[1] == max_length)
    assert(token_type.size()[0] == batch_size and token_type.size()[1] == max_length)

    return (inputs, attention, token_type), quality

def count(output, quality):
    assert output.size(0) >= len(quality)
    correct_count = 0
    count = len(quality)
```

(continues on next page)

(continued from previous page)

```
    batch_predictions = [ row.argmax().item() for row in output ]

    for a, b in zip(batch_predictions, quality):
        if int(a)==int(b):
            correct_count += 1

    return correct_count, count
```

8.23.4 Data parallel inference

In the below cell, we use the data parallel approach for inference. In this approach, we load multiple models, all of them running in parallel. Each model is loaded onto a single NeuronCore. In the below implementation, we launch 16 models, thereby utilizing all the 16 cores on an inf1.6xlarge.

> Note: Now if you try to decrease the num_cores in the above cells, please restart the notebook and run `!sudo rmmod neuron; sudo modprobe neuron` step in cell 2 to clear the Neuron cores.

Since, we can run more than 1 model concurrently, the throughput for the system goes up. To achieve maximum gain in throughput, we need to efficiently feed the models so as to keep them busy at all times. In the below setup, this is done by using a producer-consumer model. We maintain a common python queue shared across all the models. The common queue enables feeding data continuously to the models.

[ ]:
```
from parallel import NeuronSimpleDataParallel
from bert_benchmark_utils import BertTestDataset, BertResults
import time
import functools

max_length = 128
num_cores = 16
batch_size = 1

tsv_file="glue_mrpc_dev.tsv"

data_set = BertTestDataset( tsv_file=tsv_file, tokenizer=tokenizer, max_length=max_
↪length )
data_loader = torch.utils.data.DataLoader(data_set, batch_size=batch_size, shuffle=True)

#Result aggregation class (code in bert_benchmark_utils.py)
results = BertResults(batch_size, num_cores)
def result_handler(output, result_id, start, end, input_dict):
    correct_count, inference_count = count(output[0], input_dict.pop(result_id))
    elapsed = end - start
    results.add_result(correct_count, inference_count, [elapsed], [end], [start])

parallel_neuron_model = NeuronSimpleDataParallel('bert_neuron.pt', num_cores)

#Starting the inference threads
parallel_neuron_model.start_continuous_inference()

# Warm up the cores
z = torch.zeros( [batch_size, max_length], dtype=torch.long )
batch = (z, z, z)
```

(continues on next page)

(continued from previous page)

```
for _ in range(num_cores*4):
    parallel_neuron_model.infer(batch, -1, None)

input_dict = {}
input_id = 0
for _ in range(30):
    for batch in data_loader:
        batch, quality = get_input_with_padding(batch, batch_size, max_length)
        input_dict[input_id] = quality
        callback_fn = functools.partial(result_handler, input_dict=input_dict)
        parallel_neuron_model.infer(batch, input_id, callback_fn)
        input_id+=1

# Stop inference
parallel_neuron_model.stop()


with open("benchmark.txt", "w") as f:
    results.report(f, window_size=1)

with open("benchmark.txt", "r") as f:
    for line in f:
        print(line)
```

Now recompile with a larger batch size of six sentence pairs

[ ]:
```
batch_size = 6

example_inputs_paraphrase = (
    torch.cat([paraphrase['input_ids']] * batch_size,0),
    torch.cat([paraphrase['attention_mask']] * batch_size,0),
    torch.cat([paraphrase['token_type_ids']] * batch_size,0)
)

# Run torch.neuron.trace to generate a TorchScript that is optimized by AWS Neuron
model_neuron_batch = torch.neuron.trace(model, example_inputs_paraphrase)

## Save the batched model
model_neuron_batch.save('bert_neuron_b{}.pt'.format(batch_size))
```

Rerun inference with batch 6

[ ]:
```
from parallel import NeuronSimpleDataParallel
from bert_benchmark_utils import BertTestDataset, BertResults
import time
import functools

max_length = 128
num_cores = 16
batch_size = 6

data_set = BertTestDataset( tsv_file=tsv_file, tokenizer=tokenizer, max_length=max_
↪length )
```

(continues on next page)

(continued from previous page)

```
data_loader = torch.utils.data.DataLoader(data_set, batch_size=batch_size, shuffle=True)

#Result aggregation class (code in bert_benchmark_utils.py)
results = BertResults(batch_size, num_cores)
def result_handler(output, result_id, start, end, input_dict):
    correct_count, inference_count = count(output[0], input_dict.pop(result_id))
    elapsed = end - start
    results.add_result(correct_count, inference_count, [elapsed], [end], [start])

parallel_neuron_model = NeuronSimpleDataParallel('bert_neuron_b{}.pt'.format(batch_size),
↪ num_cores)

#Starting the inference threads
parallel_neuron_model.start_continuous_inference()

# Adding to the input queue to warm all cores
z = torch.zeros( [batch_size, max_length], dtype=torch.long )
batch = (z, z, z)
for _ in range(num_cores*4):
    parallel_neuron_model.infer(batch, -1, None)

#Starting the inference threads
parallel_neuron_model.start_continuous_inference()

input_dict = {}
input_id = 0
for _ in range(30):
    for batch in data_loader:
        batch, quality = get_input_with_padding(batch, batch_size, max_length)
        input_dict[input_id] = quality
        callback_fn = functools.partial(result_handler, input_dict=input_dict)
        parallel_neuron_model.infer(batch, input_id, callback_fn)
        input_id+=1

# Stop inference
parallel_neuron_model.stop()

with open("benchmark_b{}.txt".format(batch_size), "w") as f:
    results.report(f, window_size=1)

with open("benchmark_b{}.txt".format(batch_size), "r") as f:
    for line in f:
        print(line)
```

8.24 Deploy a pretrained PyTorch BERT model from HuggingFace on Amazon SageMaker with Neuron container

8.24.1 Overview

In this tutotial we will deploy on SageMaker a pretraine BERT Base model from HuggingFace Transformers, using the AWS Deep Learning Containers. We will use the same model as shown in the Neuron Tutorial "PyTorch - HuggingFace Pretrained BERT Tutorial". We will compile the model and build a custom AWS Deep Learning Container, to include the HuggingFace Transformers Library.

This Jupyter Notebook should run on a ml.c5.4xlarge SageMaker Notebook instance. You can set up your SageMaker Notebook instance by following the Get Started with Amazon SageMaker Notebook Instances documentation.

> We recommend increasing the size of the base root volume of you SM notebook instance, to accomodate the models and containers built locally. A root volume of 10Gb should suffice.

8.24.2 Install Dependencies:

This tutorial requires the following pip packages:

- torch-neuron
- neuron-cc[tensorflow]
- transformers

[ ]:
```
!pip install --upgrade --no-cache-dir torch-neuron neuron-cc[tensorflow] torchvision␣
↪torch --extra-index-url=https://pip.repos.neuron.amazonaws.com
!pip install --upgrade --no-cache-dir 'transformers==4.6.0'
```

8.24.3 Compile the model into an AWS Neuron optimized TorchScript

[ ]:
```
import torch
import torch_neuron

from transformers import AutoTokenizer, AutoModelForSequenceClassification, AutoConfig
```

[ ]:
```
# Build tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased-finetuned-mrpc")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased-finetuned-
↪mrpc", return_dict=False)

# Setup some example inputs
sequence_0 = "The company HuggingFace is based in New York City"
sequence_1 = "Apples are especially bad for your health"
sequence_2 = "HuggingFace's headquarters are situated in Manhattan"

max_length=128
paraphrase = tokenizer.encode_plus(sequence_0, sequence_2, max_length=max_length,␣
↪padding='max_length', truncation=True, return_tensors="pt")
not_paraphrase = tokenizer.encode_plus(sequence_0, sequence_1, max_length=max_length,␣
↪padding='max_length', truncation=True, return_tensors="pt")
```

(continues on next page)

(continued from previous page)

```
# Run the original PyTorch model on compilation exaple
paraphrase_classification_logits = model(**paraphrase)[0]

# Convert example inputs to a format that is compatible with TorchScript tracing
example_inputs_paraphrase = paraphrase['input_ids'], paraphrase['attention_mask'],␣
↪paraphrase['token_type_ids']
example_inputs_not_paraphrase = not_paraphrase['input_ids'], not_paraphrase['attention_
↪mask'], not_paraphrase['token_type_ids']
```

[ ]:
```
%%time
# Run torch.neuron.trace to generate a TorchScript that is optimized by AWS Neuron
# This step may need 3-5 min
model_neuron = torch.neuron.trace(model, example_inputs_paraphrase, verbose=1, compiler_
↪workdir='./compilation_artifacts')
```

You may inspect **model_neuron.graph** to see which part is running on CPU versus running on the accelerator. All native **aten** operators in the graph will be running on CPU.

[ ]:
```
# See  which part is running on CPU versus running on the accelerator.
print(model_neuron.graph)
```

Save the compiled model, so it can be packaged and sent to S3.

[ ]:
```
# Save the TorchScript for later use
model_neuron.save('neuron_compiled_model.pt')
```

Package the pre-trained model and upload it to S3

To make the model available for the SageMaker deployment, you will TAR the serialized graph and upload it to the default Amazon S3 bucket for your SageMaker session.

[ ]:
```
# Now you'll create a model.tar.gz file to be used by SageMaker endpoint
!tar -czvf model.tar.gz neuron_compiled_model.pt
```

[ ]:
```
import boto3
import time
from sagemaker.utils import name_from_base
import sagemaker
```

[ ]:
```
# upload model to S3
role = sagemaker.get_execution_role()
sess=sagemaker.Session()
region=sess.boto_region_name
bucket=sess.default_bucket()
sm_client=boto3.client('sagemaker')
```

[ ]:
```
model_key = '{}/model/model.tar.gz'.format('inf1_compiled_model')
model_path = 's3://{}/{}'.format(bucket, model_key)
boto3.resource('s3').Bucket(bucket).upload_file('model.tar.gz', model_key)
```

(continues on next page)

(continued from previous page)

```
print("Uploaded model to S3:")
print(model_path)
```

8.24.4 Build and Push the container

The following shell code shows how to build the container image using docker build and push the container image to ECR using docker push. The Dockerfile in this example is available in the **container** folder. Here's an example of the Dockerfile:

```
FROM 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference-neuron:1.7.1-neuron-
↪py36-ubuntu18.04

# Install packages
RUN pip install "transformers==4.7.0"
```

[ ]:
```
!cat container/Dockerfile
```

Before running the next cell, make sure your SageMaker IAM role has access to ECR. If not, you can attache the role `AmazonEC2ContainerRegistryPowerUser` to your IAM role ARN, which allows you to upload image layers to ECR.

It takes 5 minutes to build docker images and upload image to ECR

[ ]:
```
%%sh

# The name of our algorithm
algorithm_name=neuron-py36-inference

cd container

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none␣
↪defined)
region=$(aws configure get region)
region=${region:-us-west-2}

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.

aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1

if [ $? -ne 0 ]
then
    aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null
fi

# Get the login command from ECR in order to pull down the SageMaker PyTorch image
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-
↪stdin 763104351884.dkr.ecr.us-east-1.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
# Build the docker image locally with the image name and then push it to ECR
# with the full name.
docker build  -t ${algorithm_name} . --build-arg REGION=${region}
docker tag ${algorithm_name} ${fullname}

# Get the login command from ECR and execute it directly
aws ecr get-login-password --region ${region} | docker login --username AWS --password-
↪stdin ${account}.dkr.ecr.${region}.amazonaws.com
docker push ${fullname}
```

8.24.5 Deploy Container and run inference based on the pretrained model

To deploy a pretrained PyTorch model, you'll need to use the PyTorch estimator object to create a PyTorchModel object and set a different entry_point.

You'll use the PyTorchModel object to deploy a PyTorchPredictor. This creates a SageMaker Endpoint – a hosted prediction service that we can use to perform inference.

[ ]:
```
import sys

!{sys.executable} -m pip install Transformers
```

[ ]:
```
import os
import boto3
import sagemaker

role = sagemaker.get_execution_role()
sess = sagemaker.Session()

bucket = sess.default_bucket()
prefix = "inf1_compiled_model/model"

# Get container name in ECR
client=boto3.client('sts')
account=client.get_caller_identity()['Account']

my_session=boto3.session.Session()
region=my_session.region_name

algorithm_name="neuron-py36-inference"
ecr_image='{}.dkr.ecr.{}.amazonaws.com/{}:latest'.format(account, region, algorithm_name)
print(ecr_image)
```

An implementation of *model_fn* is required for inference script. We are going to implement our own **model_fn** and **predict_fn** for Hugging Face Bert, and use default implementations of **input_fn** and **output_fn** defined in sagemaker-pytorch-containers.

In this example, the inference script is put in **code** folder. Run the next cell to see it:

[ ]:
```
!pygmentize code/inference.py
```

Path of compiled pretrained model in S3:

[ ]:
```
key = os.path.join(prefix, "model.tar.gz")
pretrained_model_data = "s3://{}/{}".format(bucket, key)
print(pretrained_model_data)
```

The model object is defined by using the SageMaker Python SDK's PyTorchModel and pass in the model from the estimator and the entry_point. The endpoint's entry point for inference is defined by model_fn as seen in the previous code block that prints out **inference.py**. The model_fn function will load the model and required tokenizer.

Note, **image_uri** must be user's own ECR images.

[ ]:
```
from sagemaker.pytorch.model import PyTorchModel

pytorch_model = PyTorchModel(
    model_data=pretrained_model_data,
    role=role,
    source_dir="code",
    framework_version="1.7.1",
    entry_point="inference.py",
    image_uri=ecr_image
)

# Let SageMaker know that we've already compiled the model via neuron-cc
pytorch_model._is_compiled_model = True
```

The arguments to the deploy function allow us to set the number and type of instances that will be used for the Endpoint.

Here you will deploy the model to a single **ml.inf1.2xlarge** instance. It may take 6-10 min to deploy.

[ ]:
```
%%time

predictor = pytorch_model.deploy(initial_instance_count=1, instance_type="ml.inf1.2xlarge
↪")
```

[ ]:
```
print(predictor.endpoint_name)
```

Since in the input_fn we declared that the incoming requests are json-encoded, we need to use a json serializer, to encode the incoming data into a json string. Also, we declared the return content type to be json string, we Need to use a json deserializer to parse the response.

[ ]:
```
predictor.serializer = sagemaker.serializers.JSONSerializer()
predictor.deserializer = sagemaker.deserializers.JSONDeserializer()
```

Using a list of sentences, now SageMaker endpoint is invoked to get predictions.

[ ]:
```
%%time
result = predictor.predict(
    [
        "Never allow the same bug to bite you twice.",
        "The best part of Amazon SageMaker is that it makes machine learning easy.",
    ]
)
print(result)
```

[ ]:
```
%%time
result = predictor.predict(
    [
        "The company HuggingFace is based in New York City",
        "HuggingFace's headquarters are situated in Manhattan",
    ]
)
print(result)
```

8.24.6 Benchmarking your endpoint

The following cells create a load test for your endpoint. You first define some helper functions: `inference_latency` runs the endpoint request, collects cliend side latency and any errors, `random_sentence` builds random to be sent to the endpoint.

[ ]:
```
import numpy as np
import datetime
import math
import time
import boto3
import matplotlib.pyplot as plt
from joblib import Parallel, delayed
import numpy as np
from tqdm import tqdm
import random
```

[ ]:
```
def inference_latency(model,*inputs):
    """
    infetence_time is a simple method to return the latency of a model inference.

        Parameters:
            model: torch model onbject loaded using torch.jit.load
            inputs: model() args

        Returns:
            latency in seconds
    """
    error = False
    start = time.time()
    try:
        results = model(*inputs)
    except:
        error = True
        results = []
    return {'latency':time.time() - start, 'error': error, 'result': results}
```

[ ]:
```
def random_sentence():

    s_nouns = ["A dude", "My mom", "The king", "Some guy", "A cat with rabies", "A sloth
↪", "Your homie", "This cool guy my gardener met yesterday", "Superman"]
    p_nouns = ["These dudes", "Both of my moms", "All the kings of the world", "Some guys
↪", "All of a cattery's cats", "The multitude of sloths living under your bed", "Your
↪homies", "Like, these, like, all these people", "Supermen"]
```

(continues on next page)

(continued from previous page)

```
    s_verbs = ["eats", "kicks", "gives", "treats", "meets with", "creates", "hacks",
↪"configures", "spies on", "retards", "meows on", "flees from", "tries to automate",
↪"explodes"]
    p_verbs = ["eat", "kick", "give", "treat", "meet with", "create", "hack", "configure
↪", "spy on", "retard", "meow on", "flee from", "try to automate", "explode"]
    infinitives = ["to make a pie.", "for no apparent reason.", "because the sky is␣
↪green.", "for a disease.", "to be able to make toast explode.", "to know more about␣
↪archeology."]

    return (random.choice(s_nouns) + ' ' + random.choice(s_verbs) + ' ' + random.
↪choice(s_nouns).lower() or random.choice(p_nouns).lower() + ' ' + random.
↪choice(infinitives))

print([random_sentence(), random_sentence()])
```

The following cell creates `number_of_clients` concurrent threads to run `number_of_runs` requests. Once completed, a `boto3` CloudWatch client will query for the server side latency metrics for comparison.

[ ]:
```
# Defining Auxiliary variables
number_of_clients = 2
number_of_runs = 1000
t = tqdm(range(number_of_runs),position=0, leave=True)

# Starting parallel clients
cw_start = datetime.datetime.utcnow()

results = Parallel(n_jobs=number_of_clients,prefer="threads")(delayed(inference_
↪latency)(predictor.predict,[random_sentence(), random_sentence()]) for mod in t)
avg_throughput = t.total/t.format_dict['elapsed']

cw_end = datetime.datetime.utcnow()

# Computing metrics and print
latencies = [res['latency'] for res in results]
errors = [res['error'] for res in results]
error_p = sum(errors)/len(errors) *100
p50 = np.quantile(latencies[-1000:],0.50) * 1000
p90 = np.quantile(latencies[-1000:],0.95) * 1000
p95 = np.quantile(latencies[-1000:],0.99) * 1000

print(f'Avg Throughput: :{avg_throughput:.1f}\n')
print(f'50th Percentile Latency:{p50:.1f} ms')
print(f'90th Percentile Latency:{p90:.1f} ms')
print(f'95th Percentile Latency:{p95:.1f} ms\n')
print(f'Errors percentage: {error_p:.1f} %\n')

# Querying CloudWatch
print('Getting Cloudwatch:')
cloudwatch = boto3.client('cloudwatch')
statistics=['SampleCount', 'Average', 'Minimum', 'Maximum']
extended=['p50', 'p90', 'p95', 'p100']
```

(continues on next page)

(continued from previous page)

```
# Give 5 minute buffer to end
cw_end += datetime.timedelta(minutes=5)

# Period must be 1, 5, 10, 30, or multiple of 60
# Calculate closest multiple of 60 to the total elapsed time
factor = math.ceil((cw_end - cw_start).total_seconds() / 60)
period = factor * 60
print('Time elapsed: {} seconds'.format((cw_end - cw_start).total_seconds()))
print('Using period of {} seconds\n'.format(period))

cloudwatch_ready = False
# Keep polling CloudWatch metrics until datapoints are available
while not cloudwatch_ready:
  time.sleep(30)
  print('Waiting 30 seconds ...')
  # Must use default units of microseconds
  model_latency_metrics = cloudwatch.get_metric_statistics(MetricName='ModelLatency',
                                                Dimensions=[{'Name': 'EndpointName',
                                                             'Value': predictor.endpoint_
↪name},
                                                            {'Name': 'VariantName',
                                                             'Value': "AllTraffic"}],
                                                Namespace="AWS/SageMaker",
                                                StartTime=cw_start,
                                                EndTime=cw_end,
                                                Period=period,
                                                Statistics=statistics,
                                                ExtendedStatistics=extended
                                                )
  # Should be 1000
  if len(model_latency_metrics['Datapoints']) > 0:
    print('{} latency datapoints ready'.format(model_latency_metrics['Datapoints'][0][
↪'SampleCount']))
    side_avg = model_latency_metrics['Datapoints'][0]['Average'] / number_of_runs
    side_p50 = model_latency_metrics['Datapoints'][0]['ExtendedStatistics']['p50'] /␣
↪number_of_runs
    side_p90 = model_latency_metrics['Datapoints'][0]['ExtendedStatistics']['p90'] /␣
↪number_of_runs
    side_p95 = model_latency_metrics['Datapoints'][0]['ExtendedStatistics']['p95'] /␣
↪number_of_runs
    side_p100 = model_latency_metrics['Datapoints'][0]['ExtendedStatistics']['p100'] /␣
↪number_of_runs

    print(f'50th Percentile Latency:{side_p50:.1f} ms')
    print(f'90th Percentile Latency:{side_p90:.1f} ms')
    print(f'95th Percentile Latency:{side_p95:.1f} ms\n')

    cloudwatch_ready = True
```

Cleanup

Endpoints should be deleted when no longer in use, to avoid costs.

[ ]:
```
predictor.delete_endpoint(predictor.endpoint)
```

[ ]:
```

```

8.25 Transformers MarianMT Tutorial

In this tutorial, you will deploy the HuggingFace MarianMT model for text translation.

This Jupyter notebook should be run on an inf1.6xlarge instance since you will be loading and compiling several large models.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

To generate text, you will be using the beam search algorithm to incrementally generate token candidates until the full output text has been created. Unlike simple single-pass models, this algorithm divides the work into two distinct phases:

- **Encoder**: Convert the input text into an encoded representation. (Executed once)
- **Decoder**: Use the encoded representation of the input text and the current output tokens to incrementally generate the set of next best candidate tokens. (Executed many times)

In this tutorial you will perform the following steps:

- **Compile**: Compile both the Encoder and Decoder for Neuron using simplified interfaces for inference.
- **Infer**: Run on CPU and Neuron and compare results.

8.25.1 Install Dependencies:

This tutorial has the following dependencies:

- `transformers==4.0.1`
- `torch-neuron==1.7.*`
- `sentencepiece`
- `neuron-cc[tensorflow]`

The following will install the required `transformers` version. Note that encoder/decoder API changes across different minor versions requires that you are specific about the version used. Also note that the `torch-neuron` version is pinned due to `transformer` compatibility issues.

[ ]:
```
!pip install --force-reinstall --extra-index-url=https://pip.repos.neuron.amazonaws.com
↪"torch-neuron==1.7.*" "transformers==4.0.1" "protobuf<4" sentencepiece "neuron-
↪cc[tensorflow]"
```

8.25.2 Parameters

The parameters of a generative model can be tuned for different use-cases. In this example, you'll tailor the parameters to a single inference beam search for an on-demand inference use-case. See the MarianConfig for parameter details.

Rather than varying the encoder/decoder token sizes at runtime, you must define these parameters prior to compilation. The encoder/decoder token sizes are important tunable parameters as a large token sequence will offer greater sentence length flexibility but perform worse than a small token sequence.

To maximize performance on Neuron, the `num_beams`, `max_encode_length` and `max_decoder_length` should be made as small as possible for the use-case.

For this tutorial you will use a model that translates sentences of up to 32 token from English to German.

[ ]:
```
model_name = "Helsinki-NLP/opus-mt-en-de" # English -> German model
num_texts = 1                              # Number of input texts to decode
num_beams = 4                              # Number of beams per input text
max_encoder_length = 32                    # Maximum input token length
max_decoder_length = 32                    # Maximum output token length
```

8.25.3 Imports

On text generation tasks, HuggingFace Transformers defines a GenerationMixin base class which provides standard methods and algorithms to generate text. For this tutorial, you will be using the beam search algorithm on encoder/decoder architectures.

To be able to use these methods, you will be defining your own class derived from the GenerationMixin class to run a beam search. This will invoke the encoder and decoder layers in a way that is compatible with fixed sized inputs and traced modules. This means you must import the base class and the output objects (Seq2SeqLMOutput, BaseModelOutput) used by the beam_search algorithm.

[ ]:
```
import os

import torch
import numpy as np
from torch.nn import functional as F

from transformers import MarianMTModel, MarianTokenizer, MarianConfig
from transformers.generation_utils import GenerationMixin
from transformers.modeling_outputs import Seq2SeqLMOutput, BaseModelOutput
from transformers.modeling_utils import PreTrainedModel

import torch_neuron
```

8.25.4 CPU Model Execution

Start by executing the model on CPU to test its execution.

The following defines the inference function which will be used to compare the Neuron and CPU output. In this example you will display all beam search sequences that were generated. On a real on-demand use case, set the `num_return_sequences` to 1 to return only the top result.

[ ]:
```
def infer(model, tokenizer, text):

    # Truncate and pad the max length to ensure that the token size is compatible with␣
↪fixed-sized encoder (Not necessary for pure CPU execution)
    batch = tokenizer(text, max_length=max_decoder_length, truncation=True, padding='max_
↪length', return_tensors="pt")
    output = model.generate(**batch, max_length=max_decoder_length, num_beams=num_beams,␣
↪num_return_sequences=num_beams)
    results = [tokenizer.decode(t, skip_special_tokens=True) for t in output]

    print('Texts:')
    for i, summary in enumerate(results):
        print(i + 1, summary)
```

[ ]:
```
model_cpu = MarianMTModel.from_pretrained(model_name)
model_cpu.eval()

tokenizer_cpu = MarianTokenizer.from_pretrained(model_name)

sample_text = "I am a small frog."
```

[ ]:
```
infer(model_cpu, tokenizer_cpu, sample_text)
```

8.25.5 Encoder & Decoder Modules

Here you will define wrappers around the encoder and decoder portions of the generation model that are compatible with `torch.jit.trace` as well as fixed-sized inputs.

Important features which are distinct from the default configuration:

1. A fixed sized `causal_mask` in the `NeuronDecoder` rather than varying the size for each iteration. This is because Neuron requires padded `input_ids` input rather than the default behavior where this grows for each beam search iteration.
2. Disabled `return_dict`. When this is enabled, the network uses `dataclass` type outputs which are not compatible with `torch.jit.trace`.
3. Disabled `use_cache`. When this option is enabled, the network expects a collection of cache tensors which grow upon each iteration. Since Neuron requires fixed sized inputs, this must be disabled.
4. The `GenerationMixin:beam_search` implementation uses only the logits for the current iteration index from the original decoder layer output. Since inputs are padded, performance can be improved by selecting only a subset of the hidden state prior to the final linear layer. For efficiency on Neuron, this reduction (`reduce`) uses an elementwise-multiply to mask out the unused hidden values and then sums along an axis.
5. Since a reduction step is insterted between the decoder output and the final logit calculation, the original `model` attribute is not used. Instead the `NeuronDecoder` class combines the decoder, reducer, and linear layers into a combined forward pass. In the original model there is a clear distinction between the decoder layer and the final linear layer. These layers are fused together to get one large fully optimized graph.

[ ]:
```
def reduce(hidden, index):
    _, n_length, _ = hidden.shape
```

(continues on next page)

(continued from previous page)

```
    # Create selection mask
    mask = torch.arange(n_length, dtype=torch.float32) == index
    mask = mask.view(1, -1, 1)

    # Broadcast mask
    masked = torch.multiply(hidden, mask)

    # Reduce along 1st dimension
    summed = torch.sum(masked, 1)
    return torch.unsqueeze(summed, 1)


class NeuronEncoder(torch.nn.Module):

    def __init__(self, model):
        super().__init__()
        self.encoder = model.model.encoder

    def forward(self, input_ids, attention_mask):
        return self.encoder(input_ids, attention_mask=attention_mask, return_dict=False)


class NeuronDecoder(torch.nn.Module):

    def __init__(self, model, max_length):
        super().__init__()
        self.weight = model.model.shared.weight.clone().detach()
        self.bias = model.final_logits_bias.clone().detach()
        self.decoder = model.model.decoder
        self.max_length = max_length

    def forward(self, input_ids, attention_mask, encoder_outputs, index):

        # Build a fixed sized causal mask for the padded decoder input ids
        mask = np.triu(np.ones((self.max_length, self.max_length)), 1)
        mask[mask == 1] = -np.inf
        causal_mask = torch.tensor(mask, dtype=torch.float)

        # Invoke the decoder
        hidden, = self.decoder(
            input_ids=input_ids,
            encoder_hidden_states=encoder_outputs,
            encoder_padding_mask=attention_mask,
            decoder_padding_mask=None,
            decoder_causal_mask=causal_mask,
            return_dict=False,
            use_cache=False,
        )

        # Reduce decoder outputs to the specified index (current iteration)
        hidden = reduce(hidden, index)
```

(continues on next page)

(continued from previous page)

```
        # Compute final linear layer for token probabilities
        logits = F.linear(
            hidden,
            self.weight,
            bias=self.bias
        )
        return logits
```

8.25.6 GenerationMixin Class

To be able to use `GenerationMixin:beam_search` you must define your own class implementation that invokes the traced `NeuronEncoder` and `NeuronDecoder` modules. The standard generator model implementation will not work by default because it is not designed to invoke the traced models with padded inputs.

Below, the `NeuronGeneration:trace` method uses the loaded generator model and traces both the Encoder and Decoder.

Next, the following methods are copied directly from the to the original class to ensure that inference behavior is identical: - `adjust_logits_during_generation` - `_force_token_id_to_be_generated`

To invoke the Encoder and Decoder traced modules in a way that is compatible with the `GenerationMixin:beam_search` implementation, the `get_encoder`, `__call__`, and `prepare_inputs_for_generation` methods are overriden.

Lastly, the class defines methods for serialization so that the model can be easily saved and loaded.

```
[ ]: class NeuronGeneration(PreTrainedModel, GenerationMixin):

         def trace(self, model, num_texts, num_beams, max_encoder_length, max_decoder_length):
             """
             Traces the encoder and decoder modules for use on Neuron.

             This function fixes the network to the given sizes. Once the model has been
             compiled to a given size, the inputs to these networks must always be of
             fixed size.

             Args:
                 model (GenerationMixin): The transformer-type generator model to trace
                 num_texts (int): The number of input texts to translate at once
                 num_beams (int): The number of beams to computer per text
                 max_encoder_length (int): The maximum number of encoder tokens
                 max_encoder_length (int): The maximum number of decoder tokens
             """
             self.config.max_decoder_length = max_decoder_length

             # Trace the encoder
             inputs = (
                 torch.ones((num_texts, max_encoder_length), dtype=torch.long),
                 torch.ones((num_texts, max_encoder_length), dtype=torch.long),
             )
             encoder = NeuronEncoder(model)
             self.encoder = torch_neuron.trace(encoder, inputs)
```

(continues on next page)

(continued from previous page)

```
        # Trace the decoder (with expanded inputs)
        batch_size = num_texts * num_beams
        inputs = (
            torch.ones((batch_size, max_decoder_length), dtype=torch.long),
            torch.ones((batch_size, max_encoder_length), dtype=torch.long),
            torch.ones((batch_size, max_encoder_length, model.config.d_model),␣
↪dtype=torch.float),
            torch.tensor(0),
        )
        decoder = NeuronDecoder(model, max_decoder_length)
        self.decoder = torch_neuron.trace(decoder, inputs)

    # ----------------------------------------------------------------------
    # Beam Search Methods (Copied directly from transformers)
    # ----------------------------------------------------------------------

    def adjust_logits_during_generation(self, logits, cur_len, max_length):
        if cur_len == 1 and self.config.force_bos_token_to_be_generated:
            self._force_token_id_to_be_generated(logits, self.config.bos_token_id)
        elif cur_len == max_length - 1 and self.config.eos_token_id is not None:
            self._force_token_id_to_be_generated(logits, self.config.eos_token_id)
        return logits

    @staticmethod
    def _force_token_id_to_be_generated(scores, token_id) -> None:
        scores[:, [x for x in range(scores.shape[1]) if x != token_id]] = -float("inf")

    # ----------------------------------------------------------------------
    # Encoder/Decoder Invocation
    # ----------------------------------------------------------------------

    def prepare_inputs_for_generation(
        self,
        decoder_input_ids,
        encoder_outputs=None,
        attention_mask=None,
        **model_kwargs
    ):
        # Pad the inputs for Neuron
        current_length = decoder_input_ids.shape[1]
        pad_size = self.config.max_decoder_length - current_length
        return dict(
            input_ids=F.pad(decoder_input_ids, (0, pad_size)),
            attention_mask=attention_mask,
            encoder_outputs=encoder_outputs.last_hidden_state,
            current_length=torch.tensor(current_length - 1),
        )

    def get_encoder(self):
        """Helper to invoke the encoder and wrap the results in the expected structure"""
        def encode(input_ids, attention_mask, **kwargs):
            output, = self.encoder(input_ids, attention_mask)
```

(continues on next page)

(continued from previous page)

```
            return BaseModelOutput(
                last_hidden_state=output,
            )
        return encode

    def __call__(self, input_ids, attention_mask, encoder_outputs, current_length,␣
↪**kwargs):
        """Helper to invoke the decoder and wrap the results in the expected structure"""
        logits = self.decoder(input_ids, attention_mask, encoder_outputs, current_length)
        return Seq2SeqLMOutput(logits=logits)

    # ----------------------------------------------------------------------
    # Serialization
    # ----------------------------------------------------------------------

    def save_pretrained(self, directory):
        if os.path.isfile(directory):
            print(f"Provided path ({directory}) should be a directory, not a file")
            return
        os.makedirs(directory, exist_ok=True)
        torch.jit.save(self.encoder, os.path.join(directory, 'encoder.pt'))
        torch.jit.save(self.decoder, os.path.join(directory, 'decoder.pt'))
        self.config.save_pretrained(directory)

    @classmethod
    def from_pretrained(cls, directory):
        config = MarianConfig.from_pretrained(directory)
        obj = cls(config)
        obj.encoder = torch.jit.load(os.path.join(directory, 'encoder.pt'))
        obj.decoder = torch.jit.load(os.path.join(directory, 'decoder.pt'))
        return obj

    @property
    def device(self):
        return torch.device('cpu')
```

8.25.7 Execution

Using everything together from above, now the process to deploy the model is as follows:

1. Compile the model
2. Serialize an artifact
3. Load the serialized artifact
4. Execute the model Neuron

[ ]:
```
# This is the name of the folder where the artifacts will be stored on disk
neuron_name = 'NeuronMarianMT'
```

[ ]:
```
model_neuron = NeuronGeneration(model_cpu.config)

# 1. Compile the model
# Note: This may take a couple of minutes since both the encoder/decoder will be compiled
model_neuron.trace(
    model=model_cpu,
    num_texts=num_texts,
    num_beams=num_beams,
    max_encoder_length=max_encoder_length,
    max_decoder_length=max_decoder_length,
)

# 2. Serialize an artifact
# After this call you will have an `encoder.pt`, `decoder.pt` and `config.json` in the␣
↪neuron_name folder
model_neuron.save_pretrained(neuron_name)
```

[ ]:
```

# 3. Load the serialized artifact
model_neuron = NeuronGeneration.from_pretrained(neuron_name)
```

[ ]:
```
# 4. Execute the model Neuron
infer(model_neuron, tokenizer_cpu, sample_text)
```

Comparing the Neuron execution to the original CPU implementation, you will see the exact same generated text.

[ ]:
```
# CPU execution for comparison
infer(model_cpu, tokenizer_cpu, sample_text)
```

8.25.8 Appendix - BART (Mask Filling Task)

These `NeuronGeneration` class can be applied to the BART model for the task of filling in mask tokens.

[ ]:
```
from transformers import BartForConditionalGeneration, BartTokenizer
bart_name = "facebook/bart-large"
bart_model = BartForConditionalGeneration.from_pretrained(bart_name, force_bos_token_to_
↪be_generated=True)
bart_tokenizer = BartTokenizer.from_pretrained(bart_name)
bart_text = "UN Chief Says There Is No <mask> in Syria"
```

[ ]:
```
# CPU Execution
infer(bart_model, bart_tokenizer, bart_text)
```

[ ]:
```
# Neuron Execution
bart_neuron = NeuronGeneration(bart_model.config)
bart_neuron.trace(
    model=bart_model,
    num_texts=num_texts,
```

(continues on next page)

(continued from previous page)

```
    num_beams=num_beams,
    max_encoder_length=max_encoder_length,
    max_decoder_length=max_decoder_length,
)
infer(bart_neuron, bart_tokenizer, bart_text)
```

8.25.9 Appendix - Pegasus (Summarization Task)

These `NeuronGeneration` class can be applied to the Pegasus model for summarization.

```
[ ]: from transformers import PegasusForConditionalGeneration, PegasusTokenizer
pegasus_name = 'google/pegasus-xsum'
pegasus_model = PegasusForConditionalGeneration.from_pretrained(pegasus_name)
pegasus_tokenizer = PegasusTokenizer.from_pretrained(pegasus_name)
pegasus_text = "PG&E stated it scheduled the blackouts in response to forecasts for high␣
↪winds amid dry conditions. The aim is to reduce the risk of wildfires."
```

```
[ ]: # CPU Execution
infer(pegasus_model, pegasus_tokenizer, pegasus_text)
```

```
[ ]: # Neuron Execution
pegasus_neuron = NeuronGeneration(pegasus_model.config)
pegasus_neuron.trace(
    model=pegasus_model,
    num_texts=num_texts,
    num_beams=num_beams,
    max_encoder_length=max_encoder_length,
    max_decoder_length=max_decoder_length,
)
infer(pegasus_neuron, pegasus_tokenizer, pegasus_text)
```

Utilizing Neuron Capabilities Tutorials

- BERT TorchServe tutorial [html]
- NeuronCore Pipeline tutorial *[html]* [notebook]

8.26 Using NeuronCore Pipeline with PyTorch

In this tutorial you compile a pretrained BERT base model from HuggingFace Transformers, using the NeuronCore Pipeline feature of the AWS Neuron SDK. You benchmark model latency of the pipeline parallel mode and compare with the usual data parallel (multi-worker) deployment.

This tutorial is intended to run in an inf1.6xlarge, running the latest AWS Deep Learning AMI (DLAMI). The inf1.6xlarge instance size has AWS Inferentia chips for a total of 16 NeuronCores.

Verify that this Jupyter notebook is running the Python or Conda kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

Note: Do not execute this tutorial using "Run -> Run all cells" option.

8.26.1 Install Dependencies:

This tutorial requires the following pip packages:

- `torch-neuron`
- `neuron-cc[tensorflow]`
- `transformers`

Most of these packages will be installed when configuring your environment using the Neuron PyTorch setup guide. The additional HuggingFace Transformers dependency must be installed here.

[ ]:
```
!pip install --upgrade "transformers==4.6.0"
```

8.26.2 Compiling a BERT base model for a single NeuronCore

To run a HuggingFace BERTModel on Inferentia, you only need to add a single extra line of code to the usual Transformers PyTorch implementation, after importing the torch_neuron framework.

Add the argument `return_dict=False` to the BERT transformers model so it can be traced with TorchScript. TorchScript is a way to create serializable and optimizable models from PyTorch code.

Enable padding to a maximum sequence length of 128, to test the model's performance with a realistic payload size. You can adapt this sequence length to your application's requirement.

You can adapt the original example on the BertModel forward pass docstring according to the following cell

[ ]:
```
import torch
import torch_neuron
from transformers import BertTokenizer, BertModel

from joblib import Parallel, delayed
import numpy as np
from tqdm import tqdm

import os
import time


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased',return_dict=False)

inputs = tokenizer("Hello, my dog is cute",return_tensors="pt",max_length=128,padding=
↪'max_length',truncation=True)
```

The one extra line required is the call to torch.neuron.trace() method. This call compiles the model and returns the forwad method of the torch `nn.Model` method, which you can use to run inference.

The compiled graph can be saved using the `torch.jit.save` function and restored using `torch.jit.load` function for inference on Inf1 instances. During inference, the previously compiled artifacts will be loaded into the Neuron Runtime for inference execution.

```
[ ]: neuron_model = torch.neuron.trace(model,
                                       example_inputs = (inputs['input_ids'],inputs[
↪'attention_mask']),
                                       verbose=1)
```

8.26.3 Running the BERT base model on a single NeuronCore

With the model already available in memory, you can time one execution and check for the latency on the single inference call. You will load the model into Inferentia with a single inference call. A large "wall time" is expected when you first run the next cell, running the cell twice will show the actual inference latency:

```
[ ]: %%time
# The following line tests inference and should be executed on Inf1 instance family.
outputs = neuron_model(*(inputs['input_ids'],inputs['attention_mask']))
```

You can also check for the throughput of the single model running on a single NeuronCore.

The sequential inference test (for loop) does not measure all the performance one can achieve in an instance with multiple NeuronCores. To improve hardwar utilization you can run parallel inference requests over multiple model workers, which you'll test in the Data Parallel Bonus Section below.

```
[ ]: %%time
for _ in tqdm(range(100)):
    outputs = neuron_model(*(inputs['input_ids'],inputs['attention_mask']))
```

Save the compiled model for later use:

```
[ ]: neuron_model.save('bert-base-uncased-neuron.pt')
```

8.26.4 Compiling a BERT base model for 16 NeuronCores

Our next step is to compile the same model for all 16 NeuronCores available in the inf1.6xlarge and check the performance difference when running pipeline parallel inferences..

Prior to compiling and executing the model, use the following cell to restart your IPython Kernel.

> **Note:** If you run this notebook using Jupyter Notebooks, instead of Jupyterlab, you may need to restart the kernel using the "Kernel -> Restart" option, after running the next cell

```
[ ]: import IPython
# Automatically restarts kernel
IPython.Application.instance().kernel.do_shutdown(True)
```

After you get `{'status': 'ok', 'restart': True}`, reinstantiate your environment with the required libraries and the `BertTokenizer` and `BertModel` from Transformers.

```
[ ]: import torch
import torch_neuron
from transformers import BertTokenizer, BertModel

from joblib import Parallel, delayed
```

(continues on next page)

(continued from previous page)

```
import numpy as np
from tqdm import tqdm

import os
import time


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased',return_dict=False)

inputs = tokenizer("Hello, my dog is cute",return_tensors="pt",max_length=128,padding=
↪'max_length',truncation=True)
```

To enable pipeline mode during compilation, you need only to add the compiler flag `--neuroncore-pipeline-cores` and set the number of desired cores. The cell below sets up a `neuroncore_pipeline_cores` string, which you can set for the available number of NeuronCores on the instance: *inf1.6xlarge* has 16 NeuronCores in 4 Inferentia chips.

[ ]:
```
# Number of Cores in the Pipeline Mode
neuroncore_pipeline_cores = 16 # This string should be '4' on an inf1.xlarge

# Compiling for neuroncore-pipeline-cores='16'
neuron_pipeline_model = torch.neuron.trace(model,
                                           example_inputs = (inputs['input_ids'],inputs[
↪'attention_mask']),
                                           verbose=1,
                                           compiler_args = ['--neuroncore-pipeline-cores
↪', str(neuroncore_pipeline_cores)]
                                          )
```

8.26.5 Running the BERT base model on 16 NeuronCores

Next, time one execution and check for the latency on the single inference call over 16 cores. You will load the model into Inferentia with a single inference call. A large "wall time" is expected when you first run the next cell, running the cell twice will show the actual inference latency:

[ ]:
```
%%time
# The following line tests inference and should be executed on Inf1 instance family.
outputs = neuron_pipeline_model(*(inputs['input_ids'],inputs['attention_mask']))
```

Check also for the throughput of the single model running over a 16 NeuronCores.

The sequential inference test (for loop) does not measure all the performance one can achieve with Pipeline mode. As the inference runs in streaming fashion, at least 15 cores are waiting for a new call until the last one processes the first call. This results in low NeuronCore utilization. To improve hardware utilization you will require parallel inference requests, which you'll test in the next section.

[ ]:
```
for _ in tqdm(range(100)):
    outputs = neuron_pipeline_model(*(inputs['input_ids'],inputs['attention_mask']))
```

8.26.6 Load Testing the Pipeline Parallel Mode

To put the 16 NeuronCores group to test, a client has to run concurrent requests to the model. In this Notebook setup you achieve it by creating a thread pool with `Joblib.Parallel`, with all workers on the pool runing one inference call.

You can define a new method called `inference_latency()` so that you measure the amount of time each inference calls take.

[ ]:
```
def inference_latency(model,*inputs):
    """
    infetence_time is a simple method to return the latency of a model inference.

        Parameters:
            model: torch model onbject loaded using torch.jit.load
            inputs: model() args

        Returns:
            latency in seconds
    """
    start = time.time()
    _ = model(*inputs)
    return time.time() - start
```

Use `tqdm` to measure total throughput of your experiment, with a nice side-effect of "cool progress bar!". The total throughput is expected to be high, so set your experiment range to a large number, here 30k inferences.

To calculate the latency statistics over the returned 30k list of latencies use `numpy.qunatile()` method.

[ ]:
```
t = tqdm(range(30000), position=0, leave=True)
latency = Parallel(n_jobs=12,prefer="threads")(delayed(inference_latency)(neuron_
↪pipeline_model,*(inputs['input_ids'],inputs['attention_mask'])) for i in t)

p50 = np.quantile(latency[-10000:],0.50) * 1000
p95 = np.quantile(latency[-10000:],0.95) * 1000
p99 = np.quantile(latency[-10000:],0.99) * 1000
avg_throughput = t.total/t.format_dict['elapsed']
print(f'Avg Throughput: :{avg_throughput:.1f}')
print(f'50th Percentile Latency:{p50:.1f} ms')
print(f'95th Percentile Latency:{p95:.1f} ms')
print(f'99th Percentile Latency:{p99:.1f} ms')
```

Save compile model for later use:

[ ]:
```
# Save the TorchScript graph
neuron_pipeline_model.save('bert-base-uncased-neuron-pipeline.pt')
```

8.26.7 Bonus Section - Load Testing Data Parallel Mode

Prior to setting up a Data Parallel experiment using the model compiled for a single NeuronCore, run the following cell to restart your IPython Kernel.

> **Note:** If you run this notebook using Jupyter Notebooks, instead of Jupyterlab, you may need to restart the kernel using the "Kernel -> Restart" option, after running the next cell

[ ]:
```
import IPython
# Automatically restarts kernel
IPython.Application.instance().kernel.do_shutdown(True)
```

After you get `{'status': 'ok', 'restart': True}`, reinstantiate your environment with the required libraries and just the `BertTokenizer` from Transformers. You will not re-compile the model as you have the `bert-base-uncased-neuron.pt` TorchScript graph available. Also, redefine the `inference_latency()` method.

[ ]:
```
import torch
import torch_neuron
from transformers import BertTokenizer

from joblib import Parallel, delayed
import numpy as np
from tqdm import tqdm

import os
import time

def inference_latency(model,*inputs):
    """
    infetence_time is a simple method to return the latency of a model inference.

        Parameters:
            model: torch model onbject loaded using torch.jit.load
            inputs: model() args

        Returns:
            latency in seconds
    """
    start = time.time()
    _ = model(*inputs)
    return time.time() - start

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

inputs = tokenizer("Hello, my dog is cute",return_tensors="pt",max_length=128,padding=
↪'max_length',truncation=True)
```

You use the `'NEURON_RT_NUM_CORES'` environment variable to define how many Neuron cores to be used. Set the environment variable to the number of individual workers you want to test in parallel.

`torch_neuron` will load one model per NeuronCore group until it runs out of cores. At that point, if the Python process continues to spawn more model objest using `torch.jit.load`, `torch_neuron` will start stacking more than one model per core, until the Inferentia chip memory is full.

Inferentia is able to run inference over all the loaded models, but only one at a time. The Neuron Runtime takes care

of dynamically switching the model context as requests come in, no extra worker process management required. Use 1 model per NeuronCore to achieve maximum performance.

The following cell creates a list with as many models as NeuronCore Groups and execute one single dummy inference to load the models into Inferentia.

[ ]:
```
import warnings
# Number of data parallel workers
number_of_workers=16 # This number should be 4 on an inf1.xlarge

# Setting up a data parallel group
os.environ['NEURON_RT_NUM_CORES'] = str(number_of_workers)

# Loading 'number_of_workers' amount of models in Python memory
model_list = [torch.jit.load('bert-base-uncased-neuron.pt') for _ in range(number_of_
↪workers)]

# Dummy inference to load models to Inferentia
_ = [mod(*(inputs['input_ids'],inputs['attention_mask'])) for mod in model_list]
```

Adapt the call to `joblib.Parallel()` iterating over a concatenated version of the `model_list`, to run 'round-robin' calls to each of the model workers.

[ ]:
```
t = tqdm(model_list*1500,position=0, leave=True)
latency = Parallel(n_jobs=number_of_workers,prefer="threads")(delayed(inference_
↪latency)(mod,*(inputs['input_ids'],inputs['attention_mask'])) for mod in t)

p50 = np.quantile(latency[-10000:],0.50) * 1000
p95 = np.quantile(latency[-10000:],0.95) * 1000
p99 = np.quantile(latency[-10000:],0.99) * 1000
avg_throughput = t.total/t.format_dict['elapsed']
print(f'Avg Throughput: :{avg_throughput:.1f}')
print(f'50th Percentile Latency:{p50:.1f} ms')
print(f'95th Percentile Latency:{p95:.1f} ms')
print(f'99th Percentile Latency:{p99:.1f} ms')
```

For this model, despite the larger number of workers, the per-worker latency increases when running a single model per core, which in turn reduces the total throughput.

This behavior may not repeat if the model memory footprint or the input payload size changes, i.e batch size > 1. We encourage you to experiment with the data parallel and pipeline parallel modes to optimize your application performance.

Note: To use Jupyter Notebook see:

- setup-jupyter-notebook-steps-troubleshooting
- running-jupyter-notebook-as-script

Additional Examples (torch-neuron)

- AWS Neuron Samples GitHub Repository

API Reference Guide (torch-neuron)

This document is relevant for: Inf1

8.27 PyTorch-Neuron trace python API

The PyTorch-Neuron trace Python API provides a method to generate PyTorch models for execution on Inferentia, which can be serialized as TorchScript. It is analogous to `torch.jit.trace()` function in PyTorch.

`torch_neuron.trace`(*model*, *example_inputs*, ***kwargs*)

The *`torch_neuron.trace()`* method sends operations to the Neuron-Compiler (`neuron-cc`) for compilation and embeds compiled artifacts in a TorchScript graph.

Compilation can be done on any EC2 machine with sufficient memory and compute resources. c5.4xlarge or larger is recommended.

Options can be passed to Neuron compiler via the compile function. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.

This function partitions nodes into operations that are supported by Neuron and operations which are not. Operations which are not supported by Neuron are run on CPU. Graph partitioning can be controlled by the `subgraph_builder_function`, `minimum_segment_size`, and `fallback` parameters (See below). By default all supported operations are compiled and run on Neuron.

The compiled graph can be saved using the `torch.jit.save()` function and restored using `torch.jit.load()` function for inference on Inf1 instances. During inference, the previously compiled artifacts will be loaded into the Neuron Runtime for inference execution.

Required Arguments

Parameters

- **`model`** (*`Module,callable`*) – The functions that that will be run with `example_inputs` arguments. The arguments and return types must compatible with `torch.jit.trace()`. When a `Module` is passed to *`torch_neuron.trace()`*, only the `forward()` method is run and traced.
- **`example_inputs`** (*`tuple`*) – A tuple of example inputs that will be passed to the `model` while tracing. The resulting trace can be run with inputs of different types and shapes assuming the traced operations support those types and shapes. This parameter may also be a single `torch.Tensor` in which case it is automatically wrapped in a `tuple`.

Optional Keyword Arguments

Keyword Arguments

- **`compiler_args`** (*`list[str]`*) – List of strings representing `neuron-cc` compiler arguments. Note that these arguments apply to all subgraphs generated by allowlist partitioning. For example, use `compiler_args=['--neuroncore-pipeline-cores', '4']` to set number of NeuronCores per subgraph to 4. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.
- **`compiler_timeout`** (*`int`*) – Timeout in seconds for waiting `neuron-cc` to complete. Exceeding this timeout will cause a `subprocess.TimeoutExpired` exception.

- **compiler_workdir** (*str*) – Work directory used by `neuron-cc`. Useful for debugging and/or inspecting `neuron-cc` logs/IRs.
- **subgraph_builder_function** (*callable*) – A function which is evaluated on each node during graph partitioning. This takes in a torch graph operator node and returns a `bool` value of whether it should be included in the fused Neuron graph or not. By default the partitioner selects all operators which are supported by Neuron.
- **minimum_segment_size** (*int*) – A parameter used during partitioning. This specifies the minimum number of graph nodes which should be compiled into a Neuron graph (default= `2`). If the number of nodes is smaller than this size, the operations will run on CPU.
- **single_fusion_ratio_threshold** (*float*) – A parameter used during partitioning. During partitioning, if a single partition contains a fraction of operations greater than this threshold, only one graph partition will be compiled (default= `0.6`). This is used to avoid compiling many small Neuron graphs. To force compilation of all graphs to Neuron (even when they are very small), a value of `1.0` can be used.
- **fallback** (*bool*) – A function parameter to turn off graph partitioning. Indicates whether to attempt to fall back to CPU operations if an operation is not supported by Neuron. By default this is `True`. If this is set to `False` and an operation is not supported by Neuron, this will fail compilation and raise an `AttributeError`.
- **dynamic_batch_size** (*bool*) – A flag to allow Neuron graphs to consume variable sized batches of data. Dynamic sizing is restricted to the 0th dimension of a tensor.
- **optimizations** (*list*) – A list of *Optimization* passes to apply to the model.
- ****kwargs** – All other keyword arguments will be forwarded directly to `torch.jit.trace()`. This supports flags like `strict=False` in order to allow dictionary outputs.

Returns

The traced `ScriptModule` with embedded compiled neuron sub-graphs. Operations in this module will run on Neuron unless they are not supported by Neuron or manually partitioned to run on CPU.

Note that in `torch<1.8` This would return a `ScriptFunction` if the input was function type.

Return type *ScriptModule*, *ScriptFunction*

class torch_neuron.**Optimization**

A set of optimization passes that can be applied to the model.

FLOAT32_TO_FLOAT16

A post-processing pass that converts all `torch.float32` tensors to `torch.float16` tensors. The advantage to this optimization pass is that input/output tensors will be type cast. This reduces the amount of data that will be copied to and from Inferentia hardware. The resulting traced model will accept both `torch.float32` and `torch.float16` inputs where the model used `torch.float32` inputs during tracing. It is only beneficial to enable this optimization if the throughput of a model is highly dependent upon data transfer speed. This optimization is not recommended if the final application will use `torch.float32` inputs since the `torch.float16` type cast will occur on CPU during inference.

8.27.1 Example Usage

Function Compilation

```
import torch
import torch_neuron

def foo(x, y):
    return 2 * x + y

# Run `foo` with the provided inputs and record the tensor operations
traced_foo = torch.neuron.trace(foo, (torch.rand(3), torch.rand(3)))

# `traced_foo` can now be run with the TorchScript interpreter or saved
# and loaded in a Python-free environment
torch.jit.save(traced_foo, 'foo.pt')
traced_foo = torch.jit.load('foo.pt')
```

Module Compilation

```
import torch
import torch_neuron
import torch.nn as nn

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Conv2d(1, 1, 3)

    def forward(self, x):
        return self.conv(x) + 1

n = Net()
n.eval()

inputs = torch.rand(1, 1, 3, 3)

# Trace a specific method and construct `ScriptModule` with
# a single `forward` method
neuron_forward = torch.neuron.trace(n.forward, inputs)

# Trace a module (implicitly traces `forward`) and constructs a
# `ScriptModule` with a single `forward` method
neuron_net = torch.neuron.trace(n, inputs)
```

Pre-Trained Model Compilation

The following is an example usage of the compilation Python API, with default compilation arguments, using a pre-trained `torch.nn.Module`:

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)
```

Compiling models with torch.jit.trace kwargs

This example uses the `strict=False` flag to compile a model with dictionary outputs. Similarly, any other keyword argument of `torch.jit.trace()` can be passed directly to *`torch_neuron.trace()`* so that it is passed to the underlying trace call.

```
import torch
import torch_neuron
import torch.nn as nn

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv = nn.Conv2d(1, 1, 3)

    def forward(self, x):
        return {'conv': self.conv(x) + 1}

model = Model()
model.eval()

inputs = torch.rand(1, 1, 3, 3)

# use the strict=False kwarg to compile a model with dictionary outputs
# the model output format does not change
model_neuron = torch.neuron.trace(model, inputs, strict=False)
```

Dynamic Batching

This example uses the optional `dynamic_batch_size` option in order to support variable sized batches at inference time.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input of batch size 1
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image, dynamic_batch_size=True)

# Execute with a batch of 7 images
batch = torch.rand([7, 3, 224, 224])
results = model_neuron(batch)
```

Manual Partitioning

The following example uses the optional `subgraph_builder_function` parameter to ensure that only a specific convolution layer is compiled to Neuron. The remaining operations are executed on CPU.

```
import torch
import torch_neuron
import torch.nn as nn

class ExampleConvolutionLayer(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = nn.Conv2d(1, 1, 3)

    def forward(self, x):
        return self.conv(x) + 1

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer = ExampleConvolutionLayer()

    def forward(self, x):
        return self.layer(x) * 100

def subgraph_builder_function(node) -> bool:
    """Select if the node will be included in the Neuron graph"""

    # Node names are tuples of Module names.
    if 'ExampleConvolutionLayer' in node.name:
        return True
```

(continues on next page)

(continued from previous page)

```
    # Ignore all operations not in the example convolution layer
    return False

model = Model()
model.eval()

inputs = torch.rand(1, 1, 3, 3)

# Log output shows that `aten::_convolution` and `aten::add` are compiled
# but `aten::mul` is not. This will seamlessly switch between Neuron/CPU
# execution in a single graph.
neuron_model = torch_neuron.trace(
    model,
    inputs,
    subgraph_builder_function=subgraph_builder_function
)
```

This document is relevant for: Inf1

This document is relevant for: Inf1

8.28 torch.neuron.DataParallel API

The *torch.neuron.DataParallel()* Python API implements data parallelism on ScriptModule models created by the /neuron-guide/neuron-frameworks/pytorch-neuron/api-compilation-python-api.rst. This function is analogous to DataParallel in PyTorch. The *Data Parallel Inference on Torch Neuron* application note provides an overview of how *torch.neuron.DataParallel()* can be used to improve the performance of inference workloads on Inferentia.

torch.neuron.**DataParallel**(*model*, *device_ids=None*, *dim=0*)

Applies data parallelism by replicating the model on available NeuronCores and distributing data across the different NeuronCores for parallelized inference.

By default, DataParallel will use all available NeuronCores allocated for the current process for parallelism. DataParallel will apply parallelism on `dim=0` if `dim` is not specified.

DataParallel automatically enables *dynamic batching* on eligible models if `dim=0`. Dynamic batching can be dsiabled using *torch.neuron.DataParallel.disable_dynamic_batching()*. If dynamic batching is not enabled, the batch size at compilation-time must be equal to the batch size at inference-time divided by the number of NeuronCores being used. Specifically, the following must be true when dynamic batching is disabled: `input.shape[dim] / len(device_ids) == compilation_input.shape[dim]`. DataParallel will throw a warning if dynamic batching cannot be enabled.

DataParallel will try load all of a model's NEFFs onto a single NeuronCore, only if all of the NEFFs can fit on a single NeuronCore. DataParallel does not currently support models that have been compiled with *NeuronCore Pipeline*.

torch.neuron.DataParallel() requires PyTorch >= 1.8.

Required Arguments

Parameters **model** (*ScriptModule*) – Model created by the /neuron-guide/neuron-frameworks/pytorch-neuron/api-compilation-python-api.rst to be parallelized.

Optional Arguments

Parameters

- **device_ids** (*list*) – List of `int` or `'nc:#'` that specify the NeuronCores to use for parallelization (default: all NeuronCores). Refer to the *device_ids note* for a description of how `device_ids` indexing works.
- **dim** (*int*) – Dimension along which the input tensor is scattered across NeuronCores (default `dim=0`).

Attributes

Parameters

- **num_workers** (*int*) – Number of worker threads used for multithreaded inference (default: `2 * number of NeuronCores`).
- **split_size** (*int*) – Size of the input chunks (default: `max(1, input.shape[dim] // number of NeuronCores)`).

`torch.neuron.DataParallel.`**`disable_dynamic_batching`**`()`

Disables automatic dynamic batching on the DataParallel module. See *Dynamic batching disabled* for example of how DataParallel can be used with dynamic batching disabled. Use as follows:

```
>>> model_parallel = torch.neuron.DataParallel(model_neuron)
>>> model_parallel.disable_dynamic_batching()
```

Note: `device_ids` uses per-process NeuronCore granularity and zero-based indexing. Per-process granularity means that each Python process "sees" its own view of the world. Specifically, this means that `device_ids` only "sees" the NeuronCores that are allocated for the current process. Zero-based indexing means that each Python process will index its allocated NeuronCores starting at 0, regardless of the "global" index of the NeuronCores. Zero-based indexing makes it possible to redeploy the exact same code unchanged in different process. This behavior is analogous to the `device_ids` argument in the PyTorch `DataParallel` function.

As an example, assume DataParallel is run on an inf1.6xlarge, which contains four Inferentia chips each of which contains four NeuronCores:

- If `NEURON_RT_VISIBLE_CORES` is not set, a single process can access all 16 NeuronCores. Thus specifying `device_ids=["nc:0"]` will correspond to chip0:core0 and `device_ids=["nc:14"]` will correspond to chip3:core2.
- However, if two processes are launched where: process 1 has `NEURON_RT_VISIBLE_CORES=0-6` and process 2 has `NEURON_RT_VISIBLE_CORES=7-15`, `device_ids=["nc:14"]` cannot be specified in either process. Instead, chip3:core2 can only be accessed in process 2. Additionally, chip3:core2 is specified in process 2 with `device_ids=["nc:7"]`. Furthermore, in process 1, `device_ids=["nc:0"]` would correspond to chip0:core0; in process 2 `device_ids=["nc:0"]` would correspond to chip1:core3.

8.28.1 Examples

The following sections provide example usages of the *torch.neuron.DataParallel()* module.

Default usage

The default DataParallel use mode will replicate the model on all available NeuronCores in the current process. The inputs will be split on `dim=0`.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module
model_parallel = torch.neuron.DataParallel(model_neuron)

# Create a batched input
batch_size = 5
image_batched = torch.rand([batch_size, 3, 224, 224])

# Run inference with a batched input
output = model_parallel(image_batched)
```

Specifying NeuronCores

The following example uses the `device_ids` argument to use the first three NeuronCores for DataParallel inference.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module, run on the first three NeuronCores
# Equivalent to model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0,␣
↪1, 2])
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=['nc:0', 'nc:1', 'nc:
↪2'])
```

(continues on next page)

(continued from previous page)

```
# Create a batched input
batch_size = 5
image_batched = torch.rand([batch_size, 3, 224, 224])

# Run inference with a batched input
output = model_parallel(image_batched)
```

DataParallel with dim != 0

In this example we run DataParallel inference using four NeuronCores and `dim = 2`. Because `dim != 0`, dynamic batching is not enabled. Consequently, the DataParallel inference-time batch size must be four times the compile-time batch size. DataParallel will generate a warning that dynamic batching is disabled because `dim != 0`.

```
import torch
import torch_neuron

# Create an example model
class Model(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = torch.nn.Conv2d(3, 3, 3)

    def forward(self, x):
        return self.conv(x) + 1

model = Model()
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 8, 8])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module using 4 NeuronCores and dim = 2
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0, 1, 2, 3], dim=2)

# Create a batched input
# Note that image_batched.shape[dim] / len(device_ids) == image.shape[dim]
batch_size = 4 * 8
image_batched = torch.rand([1, 3, batch_size, 8])

# Run inference with a batched input
output = model_parallel(image_batched)
```

Dynamic batching

In the following example, we use the *torch.neuron.DataParallel()* module to run inference using several different batch sizes without recompiling the Neuron model.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module
model_parallel = torch.neuron.DataParallel(model_neuron)

# Create batched inputs and run inference on the same model
batch_sizes = [2, 3, 4, 5, 6]
for batch_size in batch_sizes:
    image_batched = torch.rand([batch_size, 3, 224, 224])

    # Run inference with a batched input
    output = model_parallel(image_batched)
```

Dynamic batching disabled

In the following example, we use *torch.neuron.DataParallel.disable_dynamic_batching()* to disable dynamic batching. We provide an example of a batch size that will not work when dynamic batching is disabled as well as an example of a batch size that does work when dynamic batching is disabled.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module and use 4 NeuronCores
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0, 1, 2, 3], dim=0)

# Disable dynamic batching
model_parallel.disable_dynamic_batching()
```

(continues on next page)

(continued from previous page)

```
# Create a batched input (this won't work)
batch_size = 8
image_batched = torch.rand([batch_size, 3, 224, 224])

# This will fail because dynamic batching is disabled and
# image_batched.shape[dim] / len(device_ids) != image.shape[dim]
# output = model_parallel(image_batched)

# Create a batched input (this will work)
batch_size = 4
image_batched = torch.rand([batch_size, 3, 224, 224])

# This will work because
# image_batched.shape[dim] / len(device_ids) == image.shape[dim]
output = model_parallel(image_batched)
```

Full tutorial with torch.neuron.DataParallel

For an end-to-end tutorial that uses DataParallel, see the *PyTorch Resnet Tutorial*.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.29 Experimental: NeuronCore Placement APIs

Warning: The following functionality is experimental and **will not be supported** in future releases of the NeuronSDK. This module serves only as a preview for future functionality. In future releases, equivalent functionality may be moved directly to the `torch_neuron` module and will no longer be available in the `torch_neuron.experimental` module.

Functions which enable placement of `torch.jit.ScriptModule` to specific NeuronCores. Two sets of functions are provided which can be used interchangeably but have different performance characteristics and advantages:

- The *`multicore_context()`* & *`neuron_cores_context()`* functions are context managers that allow a model to be placed on a given NeuronCore at `torch.jit.load()` time. These functions are the most efficient way of loading a model since the model is loaded directly to a NeuronCore. The alternative functions described below require that a model is unloaded from one core and then reloaded to another.
- The *`set_multicore()`* & *`set_neuron_cores()`* functions allow a model that has already been loaded to a NeuronCore to be moved to a different NeuronCore. This functionality is less efficient than directly loading a model to a NeuronCore within a context manager but allows device placement to be fully dynamic at runtime. This is analogous to the `torch.nn.Module.to()` function for device placement.

Important: A prerequisite to enable placement functionality is that the loaded `torch.jit.ScriptModule` has already been compiled with the *`torch_neuron.trace()`* API. Attempting to place a regular `torch.nn.Module` onto a NeuronCore prior to compilation will do nothing.

`torch_neuron.experimental.multicore_context()`

A context which loads all Neuron subgraphs to all visible NeuronCores.

This loads each Neuron subgraph within a `torch.jit.ScriptModule` to multiple NeuronCores without requiring multiple calls to `torch.jit.load()`. This allows a single `torch.jit.ScriptModule` to use multiple NeuronCores for concurrent threadsafe inferences. Executions use a round-robin strategy to distribute across NeuronCores.

Any calls to `torch.jit.load()` will cause any underlying Neuron subgraphs to load to the specified NeuronCores within this context. This context manager only needs to be used during the model load. After loading, inferences do not need to occur in this context in order to use the correct NeuronCores.

Note that this context is *not* threadsafe. Using multiple core placement contexts from multiple threads may not correctly place models.

Raises `RuntimeError` – If the Neuron runtime cannot be initialized.

Examples

Multiple Core Replication: Directly load a model to all visible NeuronCores. This allows a single `torch.jit.ScriptModule` to use all NeuronCores by running round-robin executions.

```
>>> with torch_neuron.experimental.multicore_context():
>>>     model = torch.jit.load('example_neuron_model.pt')
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 1
>>> model(example) # Executes on NeuronCore 2
```

`torch_neuron.experimental.neuron_cores_context(start_nc: int = - 1, nc_count: int = - 1)`

A context which sets the NeuronCore start/count for all Neuron subgraphs.

Any calls to `torch.jit.load()` will cause any underlying Neuron subgraphs to load to the specified NeuronCores within this context. This context manager only needs to be used during the model load. After loading, inferences do not need to occur in this context in order to use the correct NeuronCores.

Note that this context is *not* threadsafe. Using multiple core placement contexts from multiple threads may not correctly place models.

Parameters

- **start_nc** – The starting NeuronCore index where the Module is placed. The value `-1` automatically loads to the optimal NeuronCore (least used). Note that this index is always relative to NeuronCores visible to this process.
- **nc_count** – The number of NeuronCores to use. The value `-1` will load a model to exactly the number of cores required by that model (1 for most models, >1 when using NeuronCore Pipeline). If `nc_count` is greater than the number of NeuronCores required by the model, the model will be replicated across multiple NeuronCores. `(replications = floor(nc_count / cores_per_model))`

Raises

- `RuntimeError` – If the Neuron runtime cannot be initialized.
- `ValueError` – If the `nc_count` is an invalid number of NeuronCores.

Examples

Single Load: Directly load a model from disk to the first visible NeuronCore.

```
>>> with torch_neuron.experimental.neuron_cores_context(start_nc=0, nc_count=1):
>>>     model = torch.jit.load('example_neuron_model.pt')
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 0
```

Multiple Core Replication: Directly load a model from disk to 2 NeuronCores. This allows a single `torch.jit.ScriptModule` to use multiple NeuronCores by running round-robin executions.

```
>>> with torch_neuron.experimental.neuron_cores_context(start_nc=2, nc_count=2):
>>>     model = torch.jit.load('example_neuron_model.pt')
>>> model(example) # Executes on NeuronCore 2
>>> model(example) # Executes on NeuronCore 3
>>> model(example) # Executes on NeuronCore 2
```

Multiple Model Load: Directly load 2 models from disk and pin them to separate NeuronCores. This causes each `torch.jit.ScriptModule` to always execute on a specific NeuronCore.

```
>>> with torch_neuron.experimental.neuron_cores_context(start_nc=2):
>>>     model1 = torch.jit.load('example_neuron_model.pt')
>>> with torch_neuron.experimental.neuron_cores_context(start_nc=0):
>>>     model2 = torch.jit.load('example_neuron_model.pt')
>>> model1(example) # Executes on NeuronCore 2
>>> model1(example) # Executes on NeuronCore 2
>>> model2(example) # Executes on NeuronCore 0
>>> model2(example) # Executes on NeuronCore 0
```

`torch_neuron.experimental.`**`set_multicore`**(*trace: torch.jit.ScriptModule*)

Loads all Neuron subgraphs in a torch Module to all visible NeuronCores.

This loads each Neuron subgraph within a `torch.jit.ScriptModule` to multiple NeuronCores without requiring multiple calls to `torch.jit.load()`. This allows a single `torch.jit.ScriptModule` to use multiple NeuronCores for concurrent threadsafe inferences. Executions use a round-robin strategy to distribute across NeuronCores.

This will unload the model from an existing NeuronCore if it is already loaded.

Requires Torch 1.8+

Parameters **`trace`** – A torch module which contains one or more Neuron subgraphs.

Raises `RuntimeError` – If the Neuron runtime cannot be initialized.

Examples

Multiple Core Replication: Move a model across all visible NeuronCores after loading. This allows a single `torch.jit.ScriptModule` to use all NeuronCores by running round-robin executions.

```
>>> model = torch.jit.load('example_neuron_model.pt')
>>> torch_neuron.experimental.set_multicore(model)
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 1
>>> model(example) # Executes on NeuronCore 2
```

`torch_neuron.experimental.`**`set_neuron_cores`**(*trace: torch.jit.ScriptModule*, *start_nc: int = - 1*, *nc_count: int = - 1*)

Set the NeuronCore start/count for all Neuron subgraphs in a torch Module.

This will unload the model from an existing NeuronCore if it is already loaded.

Requires Torch 1.8+

Parameters

- **`trace`** – A torch module which contains one or more Neuron subgraphs.
- **`start_nc`** – The starting NeuronCore index where the Module is placed. The value `-1` automatically loads to the optimal NeuronCore (least used). Note that this index is always relative to NeuronCores visible to this process.
- **`nc_count`** – The number of NeuronCores to use. The value `-1` will load a model to exactly the number of cores required by that model (1 for most models, >1 when using NeuronCore Pipeline). If `nc_count` is greater than the number of NeuronCores required by the model, the model will be replicated across multiple NeuronCores. `(replications = floor(nc_count / cores_per_model))`

Raises

- **`RuntimeError`** – If the Neuron runtime cannot be initialized.
- **`ValueError`** – If the `nc_count` is an invalid number of NeuronCores.

Examples

Single Load: Move a model to the first visible NeuronCore after loading.

```
>>> model = torch.jit.load('example_neuron_model.pt')
>>> torch_neuron.experimental.set_neuron_cores(model, start_nc=0, nc_count=1)
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 0
>>> model(example) # Executes on NeuronCore 0
```

Multiple Core Replication: Replicate a model to 2 NeuronCores after loading. This allows a single `torch.jit.ScriptModule` to use multiple NeuronCores by running round-robin executions.

```
>>> model = torch.jit.load('example_neuron_model.pt')
>>> torch_neuron.experimental.set_neuron_cores(model, start_nc=2, nc_count=2)
>>> model(example) # Executes on NeuronCore 2
>>> model(example) # Executes on NeuronCore 3
>>> model(example) # Executes on NeuronCore 2
```

Multiple Model Load: Move and pin 2 models to separate NeuronCores. This causes each `torch.jit.ScriptModule` to always execute on a specific NeuronCore.

```
>>> model1 = torch.jit.load('example_neuron_model.pt')
>>> torch_neuron.experimental.set_neuron_cores(model1, start_nc=2)
>>> model2 = torch.jit.load('example_neuron_model.pt')
>>> torch_neuron.experimental.set_neuron_cores(model2, start_nc=0)
>>> model1(example) # Executes on NeuronCore 2
>>> model1(example) # Executes on NeuronCore 2
>>> model2(example) # Executes on NeuronCore 0
>>> model2(example) # Executes on NeuronCore 0
```

This document is relevant for: `Inf1`

Developer Guide (`torch-neuron`)

This document is relevant for: `Inf1`

8.30 Running inference on variable input shapes with bucketing

Table of contents

- *Introduction*
- *Applications that benefit from bucketing*
- *Implementing bucketing*
  - *Creating bucketed models*
  - *Running inference with bucketing*
- *Examples*
  - *Computer vision bucketing*
  - *End-to-end computer vision bucketing example*
  - *Natural language processing bucketing*
  - *End-to-end natural language processing bucketing example*

8.30.1 Introduction

With Inferentia, the shape of every input must be fixed at compile time. For applications that require multiple input sizes, we recommend using padding or bucketing techniques. Padding requires you to compile your model with the largest expected input size and pad every input to this maximum size. If the performance of your model using padding is not within your targets, you can consider implementing bucketing.

This guide introduces bucketing, a technique to run inference on inputs with variable shapes on Inferentia. The following sections explain how bucketing can improve the performance of inference workloads on Inferentia. It covers an overview of how bucketing works and provides examples of using bucketing in *computer vision* and *natural language processing* applications.

8.30.2 Applications that benefit from bucketing

Bucketing refers to compiling your model multiple times with different target input shapes to create "bucketed models." *Creating bucketed models* provides an overview on selecting the input shapes that you use to create bucketed models. At inference time, each input is padded until its shape matches the next largest bucket shape. The padded input is then passed into the corresponding bucketed model for inference. By compiling the same model with multiple different input shapes, the amount of input padding is reduced compared to padding every input to the maximum size in your dataset. This technique minimizes the compute overhead and improves inference performance compared to padding every image to the maximum shape in your dataset.

Bucketing works best when multiple different bucketed models are created to efficiently cover the full range of input shapes. You can fine-tune the model performance by experimenting with different bucket sizes that correspond to the distribution of input shapes in your dataset.

Bucketing can only be used if there is an upper bound on the shape of the inputs. If necessary, an upper bound on the input shape can be enforced using resizing and other forms of preprocessing.

The upper bound on the number of bucketed models that you use is dictated by the total size of the compiled bucketed models. Each Inferentia chip has 8GB of DRAM, or 2GB of DRAM per NeuronCore. An inf1.xlarge and inf1.2xlarge have 1 Inferentia chip, an inf1.6xlarge has 4 Inferentia chips, and an inf1.24xlarge has 16 Inferentia chips. Thus, you should limit the total size of all bucketed models to around 8GB per Inferentia chip or 2GB per NeuronCore. The following formula provides an approximation for the number of compiled bucketed models you can fit on each NeuronCore:

```
number-of-buckets = round(10^9 / number-of-weights-in-model)
```

We recommend using *neuron-top* to monitor the memory usage on your inf1 instance as you load multiple bucketed models.

8.30.3 Implementing bucketing

Implementing bucketing consists of two main parts: creating multiple bucketed models at compile-time and running inference using the bucketed models on (padded) inputs. The following sections describe how to implement bucketing to run inference in applications that have variable input shapes.

Creating bucketed models

Before running inference, models should be compiled for different input shapes that are representative of the input dataset. The input shapes that are used to compile the models determine the bucket shapes that are used during inference. The bucket shapes should be chosen to minimize the amount of padding on each new input. Additionally, there should always be a bucket that's large enough to handle the maximum input shape in the dataset. The limit on the number of compiled bucketed models that can be used is described in this *section*.

Running inference with bucketing

At inference time, each input should be padded to match the size of the next largest bucket, such that the height and width (or sequence length) of the padded input equals the size of the bucket. Then, the padded input should be passed into the corresponding bucket for inference. If necessary, it's important to remove and/or crop any aberrant predictions that occur in the padded region. For example, in object detection applications, bounding box predictions that occur in the padded regions should be removed to avoid erroneous predictions.

8.30.4 Examples

The following sections provide examples of applying the bucketing technique to run inference in applications that have variable input shapes.

Computer vision bucketing

As an example of implementing bucketing for computer vision models, consider an application where the height and width of images in dataset are uniformly distributed between *[400, 400]* and *[800, 800]*. Given that every input shape between *[400, 400]* and *[800, 800]* is equally likely, it could make sense to create bucketed models that divide up the range of input shapes into equally sized chunks. For example, we could create bucketed models for the input shapes *[500, 500]*, *[600, 600]*, *[700, 700]*, and *[800, 800]*.

As an example of running inference with bucketing, let's assume that we created bucketed models for the input shapes *[500, 500]*, *[600, 600]*, *[700, 700]*, and *[800, 800]*. If we receive an input with shape *[640, 640]*, we would pad the input to the next largest bucket, *[700, 700]*, and use this bucket for inference. If we receive an input with shape *[440, 540]*, we would need to pad the input to the bucket size, *[600, 600]*, and use this bucket for inference.

As another example of creating bucketed models, consider a computer vision application where the dataset is not uniformly distributed. As before, let's assume the input shapes range between *[400, 400]* to *[800, 800]*. Now, let's assume the data shape distribution is bimodal, such that *[540, 540]* and *[720, 720]* are the two most common input shapes. In this example, it might make sense to create bucketed models for input shapes *[540, 540]*, *[720, 720]*, and *[800, 800]* to target the most common shapes while still including the entire range of input shapes.

End-to-end computer vision bucketing example

In this example, we run inference in a computer vision application that has variable shaped images that range in shape from *[400, 400]* to *[800, 800]*. We create bucketed models for the input shapes *[500, 500]*, *[600, 600]*, *[700, 700]*, and *[800, 800]* to handle the variable input shapes.

```
import numpy as np
import torch
from torchvision import models
import torch_neuron

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Define the bucket sizes that will be used for compilation and inference
bucket_sizes = [(500, 500), (600, 600), (700, 700), (800, 800)]

# Create the bucketed models by compiling a model for each bucket size
buckets = {}
for bucket_size in bucket_sizes:
    # Create an example input that is the desired bucket size
    h, w = bucket_size
    image = torch.rand([1, 3, h, w])

    # Compile with the example input to create the bucketed model
    model_neuron = torch.neuron.trace(model, image)

    # Run a warm up inference to load the model into Inferentia memory
```

(continues on next page)

(continued from previous page)

```
    model_neuron(image)

    # Add the bucketed model based on its bucket size
    buckets[bucket_size] = model_neuron


def get_bucket_and_pad_image(image):
    # Determine which bucket size to use
    oh, ow = image.shape[-2:]
    target_bucket = None
    for bucket_size in bucket_sizes:
        # Choose a bucket that's larger in both the height and width dimensions
        if oh <= bucket_size[0] and ow <= bucket_size[1]:
            target_bucket = bucket_size
            break

    # Pad the image to match the size of the bucket
    h_delta = target_bucket[0] - oh
    w_delta = target_bucket[1] - ow

    b_pad = h_delta  # Bottom padding
    l_pad = 0  # Left padding
    t_pad = 0  # Top padding
    r_pad = w_delta  # Right padding

    # Pad the height and width of the image
    padding_amounts = (l_pad, r_pad, t_pad, b_pad)
    image_padded = torch.nn.functional.pad(image, padding_amounts, value=0)

    return image_padded, target_bucket


# Run inference on inputs with different shapes
for _ in range(10):
    # Create an image with a random height and width in range [400, 400] to [800, 800]
    h = int(np.random.uniform(low=400, high=800))
    w = int(np.random.uniform(low=400, high=800))
    image = torch.rand(1, 3, h, w)

    # Determine bucket and pad the image
    image_padded, target_bucket = get_bucket_and_pad_image(image)

    # Use the corresponding bucket to run inference
    output = buckets[target_bucket](image_padded)
```

Natural language processing bucketing

As an example of implementing bucketing for natural language processing models, consider an application where the lengths of tokenized sequences in a dataset are uniformly distributed between 0 and 128 tokens. Given that every tokenized sequence length between 0 and 128 is equally likely, it might make sense to create bucketed models that divide up the range of tokenized sequence lengths into equally sized chunks. For example, we could create bucketed models for tokenized sequence lengths 64 and 128.

As an example of running inference with bucketing, let's assume that we created bucketed models for the input tokenized sequence lengths 64 and 128. If we receive a tokenized sequence with length 55, we would need to pad it to the bucket size 64 and use this bucket for inference. If we receive a tokenized sequence with length 112, we would need to pad it to the bucket size 128 and use this bucket for inference.

End-to-end natural language processing bucketing example

In this example, we run inference in a natural language processing application that has variable length tokenized sequences that range from 0 to 128. We create bucketed models for lengths 64 and 128 to handle the variable input lengths.

```
import numpy as np
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch_neuron

# Build tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased-finetuned-mrpc")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased-finetuned-
↪mrpc", return_dict=False)
model.eval()

# Define the bucket sizes that will be used for compilation and inference
bucket_sizes = [64, 128]

# Create the bucketed models by compiling a model for each bucket size
buckets = {}
for bucket_size in bucket_sizes:
    # Setup some example inputs
    sequence_0 = "The company HuggingFace is based in New York City"
    sequence_1 = "HuggingFace's headquarters are situated in Manhattan"

    # Create an example input that is the desired bucket size
    paraphrase = tokenizer.encode_plus(sequence_0,
                                       sequence_1,
                                       max_length=bucket_size,
                                       padding='max_length',
                                       truncation=True,
                                       return_tensors="pt")

    # Convert example inputs to a format that is compatible with TorchScript tracing
    example_inputs_paraphrase = paraphrase['input_ids'], paraphrase['attention_mask'],␣
↪paraphrase['token_type_ids']

    # Compile with the example input to create the bucketed model
```

(continues on next page)

(continued from previous page)

```
    model_neuron = torch.neuron.trace(model, example_inputs_paraphrase)

    # Run a warm up inference to load the model into Inferentia memory
    model_neuron(*example_inputs_paraphrase)

    # Add the bucketed model based on its bucket size
    buckets[bucket_size] = model_neuron


def get_bucket_and_pad_paraphrase(paraphrase):
    # Determine which bucket size to use
    inputs = paraphrase['input_ids']
    attention = paraphrase['attention_mask']
    token_type = paraphrase['token_type_ids']
    paraphrase_len = inputs.shape[1]
    target_bucket = None
    for bucket_size in bucket_sizes:
        if paraphrase_len <= bucket_size:
            target_bucket = bucket_size
            break

    # Pad the paraphrase to match the size of the bucket
    delta = target_bucket - paraphrase_len
    zeros = torch.zeros([1, delta], dtype=torch.long)
    inputs = torch.cat([inputs, zeros], dim=1)
    attention = torch.cat([attention, zeros], dim=1)
    token_type = torch.cat([token_type, zeros], dim=1)

    paraphrase_padded = inputs, attention, token_type
    return paraphrase_padded, target_bucket


# Create two sample sequences
sequence_0 = ("The only other bear similar in size to the polar bear is the "
              "Kodiak bear, which is a subspecies of the brown bear. Adult male "
              "polar bears weigh 350-700 kg and measure 2.4-3 meters in total "
              "length. All bears are short-tailed, the polar bear's tail is "
              "relatively the shortest amongst living bears.")
sequence_1 = ("Around the Beaufort Sea, however, mature males reportedly "
              "average 450 kg. Adult females are roughly half the size of males "
              "and normally weigh 150-250 kg, measuring 1.8-2.4 meters in length. "
              "The legs are stocky and the ears and tail are small.")

# Run inference on inputs with different shapes
# We create the variable shapes by randomly cropping the sequences
for _ in range(10):
    # Get random sequence lengths between 0 and 128
    paraphrase_len = int(np.random.uniform(128))

    # Crop the paraphrase
    paraphrase_cropped = tokenizer.encode_plus(sequence_0,
                                               sequence_1,
```

(continues on next page)

(continued from previous page)

```
                                    max_length=paraphrase_len,
                                    padding='max_length',
                                    truncation=True,
                                    return_tensors="pt")

    # Determine bucket and pad the paraphrase
    paraphrase_padded, target_bucket = get_bucket_and_pad_paraphrase(paraphrase_cropped)

    # Use the corresponding bucket to run inference
    output = buckets[target_bucket](*paraphrase_padded)
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.31 Data Parallel Inference on Torch Neuron

Table of Contents

- *Introduction*
- *Data parallel inference*
- *torch.neuron.DataParallel*
  - *NeuronCore selection*
  - *Batch dim*
  - *Dynamic batching*
  - *Performance optimizations*
- *Examples*
  - *Default usage*
  - *Specifying NeuronCores*
  - *DataParallel with dim != 0*
  - *Dynamic batching*
  - *Dynamic batching disabled*
  - *Full tutorial with torch.neuron.DataParallel*

8.31.1 Introduction

This guide introduces `torch.neuron.DataParallel()`, a Python API that implements data parallelism on `ScriptModule` models created by the /neuron-guide/neuron-frameworks/pytorch-neuron/api-compilation-python-api.rst. The following sections explain how data parallelism can improve the performance of inference workloads on Inferentia, including how `torch.neuron.DataParallel()` uses dynamic batching to run inference on variable input sizes. It covers an overview of the `torch.neuron.DataParallel()` module and provides a few *example data parallel applications*.

8.31.2 Data parallel inference

Data Parallelism is a form of parallelization across multiple devices or cores, referred to as nodes. Each node contains the same model and parameters, but data is distributed across the different nodes. By distributing the data across multiple nodes, data parallelism reduces the total execution time of large batch size inputs compared to sequential execution. Data parallelism works best for smaller models in latency sensitive applications that have large batch size requirements.

8.31.3 torch.neuron.DataParallel

To fully leverage the Inferentia hardware, we want to use all available NeuronCores. An inf1.xlarge and inf1.2xlarge have four NeuronCores, an inf1.6xlarge has 16 NeuronCores, and an inf1.24xlarge has 64 NeuronCores. For maximum performance on Inferentia hardware, we can use `torch.neuron.DataParallel()` to utilize all available NeuronCores.

`torch.neuron.DataParallel()` implements data parallelism at the module level by replicating the Neuron model on all available NeuronCores and distributing data across the different cores for parallelized inference. This function is analogous to `DataParallel` in PyTorch. `torch.neuron.DataParallel()` requires PyTorch >= 1.8.

The following sections provide an overview of some of the features of `torch.neuron.DataParallel()` that enable maximum performance on Inferentia.

NeuronCore selection

By default, DataParallel will try to use all NeuronCores allocated to the current process to fully saturate the Inferentia hardware for maximum performance. It is more efficient to make the batch dimension divisible by the number of NeuronCores. This will ensure that NeuronCores are not left idle during parallel inference and the Inferentia hardware is fully utilized.

In some applications, it is advantageous to use a subset of the available NeuronCores for DataParallel inference. DataParallel has a `device_ids` argument that accepts a list of `int` or `'nc:#'` that specify the NeuronCores to use for parallelization. See *Specifying NeuronCores* for an example of how to use `device_ids` argument.

Batch dim

DataParallel accepts a `dim` argument that denotes the batch dimension used to split the input data for distributed inference. By default, DataParalell splits the inputs on `dim = 0` if the `dim` argument is not specified. For applications with a non-zero batch dim, the `dim` argument can be used to specify the inference-time input batch dimension. *DataParallel with dim ! = 0* provides an example of data parallel inference on inputs with batch dim = 2.

Dynamic batching

Batch size has a direct impact on model performance. The Inferentia chip is optimized to run with small batch sizes. This means that a Neuron compiled model can outperform a GPU model, even if running single digit batch sizes.

As a general best practice, we recommend optimizing your model's throughput by compiling the model with a small batch size and gradually increasing it to find the peak throughput on Inferentia.

Dynamic batching is a feature that allows you to use tensor batch sizes that the Neuron model was not originally compiled against. This is necessary because the underlying Inferentia hardware will always execute inferences with the batch size used during compilation. Fixed batch size execution allows tuning the input batch size for optimal performance. For example, batch size 1 may be best suited for an ultra-low latency on-demand inference application, while batch size > 1 can be used to maximize throughput for offline inferencing. Dynamic batching is implemented by slicing large input tensors into chunks that match the batch size used during the `torch_neuron.trace()` compilation call.

The `torch.neuron.DataParallel()` class automatically enables dynamic batching on eligible models. This allows us to run inference in applications that have inputs with a variable batch size without needing to recompile the model. See *Dynamic batching* for an example of how DataParallel can be used to run inference on inputs with a dynamic batch size without needing to recompile the model.

Dynamic batching using small batch sizes can result in sub-optimal throughput because it involves slicing tensors into chunks and iteratively sending data to the hardware. Using a larger batch size at compilation time can use the Inferentia hardware more efficiently in order to maximize throughput. You can test the tradeoff between individual request latency and total throughput by fine-tuning the input batch size.

Automatic batching in the DataParallel module can be disabled using the `disable_dynamic_batching()` function as follows:

```
>>> model_parallel = torch.neuron.DataParallel(model_neuron)
>>> model_parallel.disable_dynamic_batching()
```

If dynamic batching is disabled, the compile-time batch size must be equal to the inference-time batch size divided by the number of NeuronCores. *DataParallel with dim != 0* and *Dynamic batching disabled* provide examples of running DataParallel inference with dynamic batching disabled.

Performance optimizations

The DataParallel module has a `num_workers` attribute that can be used to specify the number of worker threads used for multithreaded inference. By default, `num_workers = 2 * number of NeuronCores`. This value can be fine tuned to optimize DataParallel performance.

DataParallel has a `split_size` attribute that dictates the size of the input chunks that are distributed to each NeuronCore. By default, `split_size = max(1, input.shape[dim] // number of NeuronCores)`. This value can be modified to optimally match the inference input chunk size with the compile-time batch size.

8.31.4 Examples

The following sections provide example usages of the *torch.neuron.DataParallel()* module.

Default usage

The default DataParallel use mode will replicate the model on all available NeuronCores in the current process. The inputs will be split on `dim=0`.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module
model_parallel = torch.neuron.DataParallel(model_neuron)

# Create a batched input
batch_size = 5
image_batched = torch.rand([batch_size, 3, 224, 224])

# Run inference with a batched input
output = model_parallel(image_batched)
```

Specifying NeuronCores

The following example uses the `device_ids` argument to use the first three NeuronCores for DataParallel inference.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module, run on the first three NeuronCores
# Equivalent to model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0,␣
↪1, 2])
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=['nc:0', 'nc:1', 'nc:
↪2'])
```

(continues on next page)

(continued from previous page)

```
# Create a batched input
batch_size = 5
image_batched = torch.rand([batch_size, 3, 224, 224])

# Run inference with a batched input
output = model_parallel(image_batched)
```

DataParallel with dim != 0

In this example we run DataParallel inference using four NeuronCores and `dim = 2`. Because `dim != 0`, dynamic batching is not enabled. Consequently, the DataParallel inference-time batch size must be four times the compile-time batch size. DataParallel will generate a warning that dynamic batching is disabled because `dim != 0`.

```
import torch
import torch_neuron

# Create an example model
class Model(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = torch.nn.Conv2d(3, 3, 3)

    def forward(self, x):
        return self.conv(x) + 1

model = Model()
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 8, 8])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module using 4 NeuronCores and dim = 2
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0, 1, 2, 3], dim=2)

# Create a batched input
# Note that image_batched.shape[dim] / len(device_ids) == image.shape[dim]
batch_size = 4 * 8
image_batched = torch.rand([1, 3, batch_size, 8])

# Run inference with a batched input
output = model_parallel(image_batched)
```

Dynamic batching

In the following example, we use the *torch.neuron.DataParallel()* module to run inference using several different batch sizes without recompiling the Neuron model.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module
model_parallel = torch.neuron.DataParallel(model_neuron)

# Create batched inputs and run inference on the same model
batch_sizes = [2, 3, 4, 5, 6]
for batch_size in batch_sizes:
    image_batched = torch.rand([batch_size, 3, 224, 224])

    # Run inference with a batched input
    output = model_parallel(image_batched)
```

Dynamic batching disabled

In the following example, we use *torch.neuron.DataParallel.disable_dynamic_batching()* to disable dynamic batching. We provide an example of a batch size that will not work when dynamic batching is disabled as well as an example of a batch size that does work when dynamic batching is disabled.

```
import torch
import torch_neuron
from torchvision import models

# Load the model and set it to evaluation mode
model = models.resnet50(pretrained=True)
model.eval()

# Compile with an example input
image = torch.rand([1, 3, 224, 224])
model_neuron = torch.neuron.trace(model, image)

# Create the DataParallel module and use 4 NeuronCores
model_parallel = torch.neuron.DataParallel(model_neuron, device_ids=[0, 1, 2, 3], dim=0)

# Disable dynamic batching
model_parallel.disable_dynamic_batching()
```

(continues on next page)

(continued from previous page)

```
# Create a batched input (this won't work)
batch_size = 8
image_batched = torch.rand([batch_size, 3, 224, 224])

# This will fail because dynamic batching is disabled and
# image_batched.shape[dim] / len(device_ids) != image.shape[dim]
# output = model_parallel(image_batched)

# Create a batched input (this will work)
batch_size = 4
image_batched = torch.rand([batch_size, 3, 224, 224])

# This will work because
# image_batched.shape[dim] / len(device_ids) == image.shape[dim]
output = model_parallel(image_batched)
```

Full tutorial with torch.neuron.DataParallel

For an end-to-end tutorial that uses DataParallel, see the *PyTorch Resnet Tutorial*.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.32 Developer Guide - PyTorch Neuron (`torch-neuron`) `LSTM` Support

The *torch-neuron* package can support `LSTM` operations and yield high performance on both fixed-length and variable-length sequences. Most network configurations can be supported, with the exception of those that require `PackedSequence` usage outside of `LSTM` or `pad_packed_sequence()` operations. Neuron must guarantee that the shapes can remain fixed throughout the network.

The following sections describe which scenarios can and cannot be supported.

8.32.1 Supported Usage

Fixed-Length Sequences

In normal usage of an `LSTM`, the inputs and outputs are expected to be a fixed size sequence length. This is the most basic usage of an `LSTM` but may not be applicable to applications where the input sequence length may vary.

```
import torch
import torch_neuron

class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs):
```

(continues on next page)

(continued from previous page)

```
        output, (ht, ct) = self.lstm(inputs)
        return output, (ht, ct)

# Example Inputs
seq_len, batch_size, input_size = 5, 2, 3
inputs = torch.rand(seq_len, batch_size, input_size)

# Trace
torch_neuron.trace(Network(), (inputs,))
```

Packed Input, Padded Output, *Pre-Sorted* Inputs

A common usage of an `LSTM` is when the input sequence sizes vary according to an input sequence lengths (such as tokens).

For example, the following sentences could result in two different sequence lengths after tokenization:

```
# Input
text = [
   'Hello, sailor',
   'Example',
]

# ... Tokenization ...

# Result
tokens = [
    [101, 7592, 1010, 11803, 102],
    [101, 2742,  102,     0,   0],
]
lengths = [5, 3]
```

Because the lengths are different, the final `LSTM` state will be dependent upon the lengths of each sequence in the batch. Torch provides a way to deal with these types of sequences by densely packing batches into a `PackedSequence`. The most common way this is constructed is by using the `pack_padded_sequence()` utility function prior to feeding inputs into the `LSTM`.

Packing the above sequences would result in the following data and batch size tensors.

```
data = [101, 101, 7592, 2742, 1010, 102, 11803, 102]
batch_sizes = [2, 2, 2, 1, 1]
```

In addition to correctly computing final `LSTM` state, using a packed sequence instead of a padded sequence also improves model performance on CPU. On Neuron, where computation is fixed to the maximum length ahead of time, **this is does not improve performance**.

When an `LSTM` is processing a `PackedSequence`, it must do so in a descending sorted length order. To ensure that sequences are sorted, `pack_padded_sequence()` provides an `enforce_sorted` flag. When `enforce_sorted` is `True`, the input is *already expected* to contain sequences sorted by length in a decreasing order along the batch dimension. Note that this must be enforced in the application-level code but is only relevant when batch size > 1.

The following network can compile successfully because the input and output to the network are guaranteed to be a fixed shape. The input shape is expected to be a padded tensor and the output tensor is expected to be padded to the maximum sequence length using the `pad_packed_sequence()` function call:

```
import torch
import torch_neuron

class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs, lengths):
        packed_input = torch.nn.utils.rnn.pack_padded_sequence(
            inputs,
            lengths=lengths,
            enforce_sorted=True,
        )
        packed_result, (ht, ct) = self.lstm(packed_input)
        padded_result, _ = torch.nn.utils.rnn.pad_packed_sequence(packed_result)
        return padded_result, ht, ct

# Example Inputs
seq_len, batch_size, input_size = 5, 2, 3
inputs = torch.rand(seq_len, batch_size, input_size)
lengths = torch.tensor([seq_len] * batch_size)

# Trace
torch_neuron.trace(Network(), (inputs, lengths))
```

Packed Input, Padded Output, *Unsorted* Inputs

When `enforce_sorted` is `False`, the input will be sorted unconditionally. This causes some CPU overhead on Neuron because unsupported operators will be inserted into the graph such as `aten::sort` and `aten::scatter_`. The `aten::lstm` operation can still be supported, but it will be less efficient than when `enforce_sorted` is `True`.

The following code is able to be traced, but results in the sorting operations running on CPU. This is not problematic in this case because the `aten::sort` and `aten::scatter_` are executed on CPU at the very beginning of the graph just prior to Neuron execution.

Like the previous example, the call to `pad_packed_sequence()` ensures that the output is a fixed-shape based on the maximum sequence length.

```
import torch
import torch_neuron

class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs, lengths):
        packed_input = torch.nn.utils.rnn.pack_padded_sequence(
            inputs,
            lengths=lengths,
```

(continues on next page)

(continued from previous page)

```
            enforce_sorted=False,
        )
        packed_result, (ht, ct) = self.lstm(packed_input)
        padded_result, _ = torch.nn.utils.rnn.pad_packed_sequence(packed_result)
        return padded_result, ht, ct

# Example Inputs
seq_len, batch_size, input_size = 5, 2, 3
inputs = torch.rand(seq_len, batch_size, input_size)
lengths = torch.tensor([seq_len] * batch_size)

# Trace
trace = torch_neuron.trace(Network(), (inputs, lengths))
```

Packed Inputs, Final Hidden & Cell State Only

When **only** the final `LSTM` hidden & cell state is used, it does not matter if the inputs are packed or unpacked since these state tensors will not vary in size.

```
import torch
import torch_neuron

class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs, lengths):
        packed_input = torch.nn.utils.rnn.pack_padded_sequence(
            inputs,
            lengths=lengths,
            enforce_sorted=True,
        )
        packed_output, (ht, ct) = self.lstm(packed_input)
        return ht, ct

# Example Inputs
seq_len, batch_size, input_size = 5, 2, 3
inputs = torch.rand(seq_len, batch_size, input_size)
lengths = torch.tensor([seq_len] * batch_size)

# Trace
trace = torch_neuron.trace(Network(), (inputs, lengths))
```

Note that when the `packed_output` is unused, it does not need to be passed to the `pad_packed_sequence()` to enable the `LSTM` to be compiled.

8.32.2 Unsupported Usage

Neuron does not support the use of a `PackedSequence` outside of the `LSTM` operation and the `pad_packed_sequence()` operation. This is because the shape of a `PackedSequence` can vary depending on the input data. This is incompatible with the Neuron restriction that all tensor sizes must be known at compilation time. When a `PackedSequence` is used only by an `LSTM` or `pad_packed_sequence()` operation, Neuron *can guarantee* the size of the intermediary tensors by padding on behalf of the application.

This means that If the `PackedSequence` is either used by a different operation or returned from the network this would result in all of the `LSTM` operations to be executed on CPU or the network compilation will fail.

`PackedSequence` Returned

The following is unsupported because the `PackedSequence` result of the `LSTM` is returned by the network:

```
class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs, lengths):
        packed_input = torch.nn.utils.rnn.pack_padded_sequence(
            inputs,
            lengths=lengths,
            enforce_sorted=False,
        )
        packed_result, (ht, ct) = self.lstm(packed_input)
        return packed_result.data, ht, ct
```

Behavior: In this case, compilation fails and the following warning is generated:

```
Operator "aten::lstm" consuming a PackedSequence input can only be supported when its␣
↪corresponding PackedSequence output is unused or unpacked using "aten::_pad_packed_
↪input". Found usage by "prim::Return"
```

Resolution: To avoid this error, the `packed_result` should be padded prior to being returned from the network by using `pad_packed_sequence()`

Invalid `PackedSequence` Usage

The following is unsupported because the `PackedSequence` result of the `LSTM` is used by a non-LSTM operator:

```
class Network(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.lstm = torch.nn.LSTM(input_size=3, hidden_size=7)

    def forward(self, inputs, lengths):
        packed_input = torch.nn.utils.rnn.pack_padded_sequence(
            inputs,
            lengths=lengths,
```

(continues on next page)

(continued from previous page)

```
            enforce_sorted=False,
        )
        packed_result, (ht, ct) = self.lstm(packed_input)
        return torch.max(packed_result.data)
```

Behavior: In this case, compilation fails and the following warning is generated:

```
Operator "aten::lstm" consuming a PackedSequence input can only be supported when its␣
↪corresponding PackedSequence output is unused or unpacked using "aten::_pad_packed_
↪input". Found usage by "aten::max"
```

Resolution: To avoid this error, the `packed_result` should be padded prior to being used in the `max()` from the network by using `pad_packed_sequence()`.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.33 Developer Guide - PyTorch Neuron (`torch-neuron`) Core Placement

This programming guide describes the available techniques and APIs to be able to allocate NeuronCores to a process and place models onto specific NeuronCores. In order of precedence, the current recommendation is to use the following placement techniques:

1. For most regular models, default core placement should be used in conjunction with `NEURON_RT_NUM_CORES` (*Default Core Allocation & Placement*)
2. For more specific core placement for NeuronCore Pipelined models, then `NEURONCORE_GROUP_SIZES` should be used (*NEURONCORE_GROUP_SIZES*).
3. Finally, for even more granular control, then the experimental explicit placement APIs may be used (*Experimental: Explicit Core Placement*).

Table of Contents

- *Developer Guide - PyTorch Neuron (`torch-neuron`) Core Placement*
  - *NeuronCore Pipeline*
  - *Default Core Allocation & Placement*
    - *Example: Default*
    - *Example: `NEURON_RT_NUM_CORES`*
    - *Example: `NEURON_RT_VISIBLE_CORES`*
    - *Example: Overlapping Models*
    - *Example: Multiple Processes*
  - *NEURONCORE_GROUP_SIZES*
    - *Example: Single NeuronCore Group*
    - *Example: Multiple NeuronCore Groups*

    * *Issue: Overlapping Models with Differing Model Sizes*
    * *Issue: Incompatible Model Sizes*
    * *Issue: Multiple Model Copies*
    * *Issue Summary*
  - *Experimental: Explicit Core Placement*
    * *Example: Manual Core Selection*
    * *Example: Automatic Multicore*
    * *Example: Explicit Replication*

The following guide will assume a machine with 8 NeuronCores:

- NeuronCores will use the notation `nc0`, `nc1`, etc.
- NeuronCore Groups will use the notation `ncg0`, `ncg1` etc.
- Models will use the notation `m0`, `m1` etc.

NeuronCores, NeuronCore Groups, and model allocations will be displayed in the following format:

Note that the actual cores that are visible to the process can be adjusted according to the *Neuron Runtime Configuration*.

8.33.1 NeuronCore Pipeline

A key concept to understand the intent behind certain core placement strategies is NeuronCore Pipelining (See *NeuronCore Pipeline*). NeuronCore Pipelining allows a model to be automatically split into pieces and executed on different NeuronCores.

For most models only 1 NeuronCore will be required for execution. A model will **only** require more than one NeuronCore when using NeuronCore Pipeline. When model pipelining is enabled, the model is split between multiple NeuronCores and data is transferred between them. For example, if the compiler flag `--neuroncore-pipeline-cores 4` is used, this splits the model into 4 pieces to be executed on 4 separate NeuronCores.

8.33.2 Default Core Allocation & Placement

The most basic requirement of an inference application is to be able to place a single model on a single NeuronCore. More complex applications may use multiple NeuronCores or even multiple processes each executing different models. The important thing to note about designing an inference application is that a single NeuronCore will always be allocated to a single process. *Processes do not share NeuronCores*. Different configurations can be used to ensure that an application process has enough NeuronCores allocated to execute its model(s):

- Default: A process will attempt to take ownership of **all NeuronCores** visible on the instance. This should be used when an instance is only running a single inference process since no other process will be allowed to take ownership of any NeuronCores.
- `NEURON_RT_NUM_CORES`: Specify the **number of NeuronCores** to allocate to the process. This places no restrictions on which NeuronCores will be used, however, the resulting NeuronCores will always be contiguous. This should be used in multi-process applications where each process should only use a subset of NeuronCores.
- `NEURON_RT_VISIBLE_CORES`: Specifies exactly **which NeuronCores** are allocated to the process by index. Similar to `NEURON_RT_NUM_CORES`, this can be used in multi-process applications where each process should only use a subset of NeuronCores. This provides more fined-grained controls over the exact NeuronCores that are allocated to a given process.

- NEURONCORE_GROUP_SIZES: Specifies a number of **NeuronCore Groups** which are allocated to the process. This is described in more detail in the *NEURONCORE_GROUP_SIZES* section.

See the *Neuron Runtime Configuration* for more environment variable details.

Example: Default

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
m1 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc1
```

With no environment configuration, the process will take ownership of all NeuronCores. In this example, only two of the NeuronCores are used by the process and the remaining are allocated but left idle.

Example: NEURON_RT_NUM_CORES

Environment Setup:

```
export NEURON_RT_NUM_CORES = '2'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
m1 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc1
```

Since there is no other process on the instance, only the first 2 NeuronCores will be acquired by the process. Models load in a simple linear order to the least used NeuronCores.

Example: NEURON_RT_VISIBLE_CORES

Environment Setup:

```
export NEURON_RT_VISIBLE_CORES = '4-5'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc4
m1 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc5
```

Unlike NEURON_RT_NUM_CORES, setting the visible NeuronCores allows the process to take control of a specific contiguous set. This allows an application to have a more fine-grained control of where models will be placed.

Example: Overlapping Models

Environment Setup:

```
export NEURON_RT_VISIBLE_CORES = '0-1'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
m1 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc0-nc1
m2 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc1
```

This shows how models may share NeuronCores but the default model placement will attempt to evenly distribute NeuronCore usage rather than overlapping all models on a single NeuronCore.

Example: Multiple Processes

Environment Setup:

```
export NEURON_RT_NUM_CORES = '2'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
m1 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc1
```

In this example, if the script is run **twice**, the following allocations will be made:

Note that each process will take ownership of as many NeuronCores as is specified by the `NEURON_RT_NUM_CORES` configuration.

8.33.3 NEURONCORE_GROUP_SIZES

Important: The use of explicit core placement should only be used when a specific performance goal is required. By default `torch-neuron` places models on the **least used** NeuronCores. This should be optimal for most applications.

Secondly, `NEURONCORE_GROUP_SIZES` is being deprecated in a future release and should be avoided in favor of newer placement methods. Use `NEURON_RT_NUM_CORES` or `NEURON_RT_VISIBLE_CORES` with default placement if possible (See *Default Core Allocation & Placement*)

In the current release of NeuronSDK, the most well-supported method of placing models onto specific NeuronCores is to use the `NEURONCORE_GROUP_SIZES` environment variable. This will define a set of "NeuronCore Groups" for the application process.

NeuronCore Groups are *contiguous sets of NeuronCores* that are allocated to a given process. Creating groups allows an application to ensure that a model has a defined set of NeuronCores that will always be allocated to it.

Note that NeuronCore Groups *can* be used to allocate non-pipelined models (those requiring exactly 1 NeuronCore) to specific NeuronCores but this is not the primary intended use. The intended use of NeuronCore Groups is to ensure pipelined models (those requiring >1 NeuronCore) have exclusive access to a specific set of contiguous NeuronCores.

In the cases where models are being used *without* NeuronCore Pipeline, the general recommendation is to use default placement (See *Default Core Allocation & Placement*).

The following section demonstrates how `NEURONCORE_GROUP_SIZES` can be used and the issues that may arise.

Example: Single NeuronCore Group

In the example where one model requires 4 NeuronCores, the correct environment configuration would be:

Environment Setup:

```
export NEURONCORE_GROUP_SIZES = '4'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-4-neuron-pipeline-cores.pt')  # Loads to nc0-nc3
```

This is the most basic usage of a NeuronCore Group. The environment setup causes the process to take control of 4 NeuronCores and then the script loads a model compiled with a NeuronCore Pipeline size of 4 to the first group.

Example: Multiple NeuronCore Groups

With more complicated configurations, the intended use of `NEURONCORE_GROUP_SIZES` is to create 1 Group per model with the correct size to ensure that the models are placed on the intended NeuronCores. Similarly, the environment would need to be configured to create a NeuronCore Group for each model:

Environment Setup:

```
export NEURONCORE_GROUP_SIZES = '3,4,1'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-3-neuron-pipeline-cores.pt')  # Loads to nc0-nc2
m1 = torch.jit.load('model-with-4-neuron-pipeline-cores.pt')  # Loads to nc3-nc6
m2 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc7
```

Issue: Overlapping Models with Differing Model Sizes

When multiple models are loaded to a single NeuronCore Group, this can cause unintended inefficiencies. A single model is only intended to span a single NeuronCore Group. Applications with many models of varying sizes can be restricted by NeuronCore Group configurations since the most optimal model layout may require more fine-grained controls.

Environment Setup:

```
export NEURONCORE_GROUP_SIZES = '2,2'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc0-nc1
m1 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc2-nc3
m2 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
m3 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc2
m4 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc0
```

Here the `NEURONCORE_GROUP_SIZES` does not generate an optimal layout because placement strictly follows the layout of NeuronCore Groups. A potentially more optimal layout would be to place `m4` onto `nc1`. In this case, since a pipelined model will not be able to have exclusive access to a set of NeuronCores, the default NeuronCore placement (no NeuronCore Groups specified) would more evenly distribute the models.

Also note here that this is an example of where the order of model loads affects which model is assigned to which NeuronCore Group. If the order of the load statements is changed, models may be assigned to different NeuronCore Groups.

Issue: Incompatible Model Sizes

Another problem occurs when attempting to place a model which does not evenly fit into a single group:

Environment Setup:

```
export NEURONCORE_GROUP_SIZES = '2,2'
```

Python Script:

```
import torch
import torch_neuron

m0 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc0-nc1
m1 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc2-nc3
m2 = torch.jit.load('model-with-3-neuron-pipeline-cores.pt')  # Loads to nc0-nc2
```

The model will be placed *across* NeuronCore Groups since there is no obvious group to assign the model to according to the environment variable configuration. Depending on the individual model and application requirements, the placement here may not be optimal.

Issue: Multiple Model Copies

It is common in inference serving applications to use multiple replicas of a single model across different NeuronCores. This allows the hardware to be fully utilized to maximize throughput. In this scenario, when using NeuronCore Groups, the only way to replicate a model on multiple NeuronCores is to create a *new model* object. In the example below, 4 models loads are performed to place a model in each NeuronCore Group.

Environment Setup:

```
export NEURONCORE_GROUP_SIZES = '2,2,2,2'
```

Python Script:

```
import torch
import torch_neuron

models = list()
for _ in range(4):
    model = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')
    models.append(model)
```

The largest consequence of this type of model allocation is that the application code is responsible for routing inference requests to models. There are a variety of ways to implement the inference switching but in all cases routing logic needs to be implemented in the application code.

Issue Summary

The use of `NEURONCORE_GROUP_SIZES` has the following problems:

- **Variable Sized Models**: Models which require crossing NeuronCore Group boundaries may be placed poorly. This means group configuration limits the size of which models can be loaded.
- **Model Load Order**: Models are loaded to NeuronCore Groups greedily. This means that the order of model loads can potentially negatively affect application performance by causing unintentional overlap.
- **Implicit Placement**: NeuronCore Groups cannot be explicitly chosen in the application code.
- **Manual Replication**: When loading multiple copies of a model to different NeuronCore Groups, this requires that multiple model handles are used.

8.33.4 Experimental: Explicit Core Placement

To address the limitations of `NEURONCORE_GROUP_SIZES`, a new set of APIs has been added which allows specific NeuronCores to be chosen by the application code. These can be found in the *Experimental: NeuronCore Placement APIs* documentation.

Example: Manual Core Selection

The most direct usage of the placement APIs is to manually select the start NeuronCore that each model is loaded to. This will automatically use as many NeuronCores as is necessary for that model (1 for most models, >1 for NeuronCore Pipelines models).

Environment Setup:

```
export NEURON_RT_NUM_CORES = '4'
```

Python Script:

```
import torch
import torch_neuron

# NOTE: Order of loads does NOT matter

with torch_neuron.experimental.neuron_cores_context(2):
    m1 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc2-nc3

with torch_neuron.experimental.neuron_cores_context(0):
    m2 = torch.jit.load('model-with-3-neuron-pipeline-cores.pt')  # Loads to nc0-nc2

with torch_neuron.experimental.neuron_cores_context(0):
    m0 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads to nc0-nc1

with torch_neuron.experimental.neuron_cores_context(3):
    m3 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads to nc3
```

Note that this directly solves the `NEURONCORE_GROUP_SIZES` issues of:

- **Variable Sized Models**: Now since models are directly placed on the NeuronCores requested by the application, there is no disconnect between the model sizes and NeuronCore Group sizes.
- **Model Load Order**: Since the NeuronCores are explicitly selected, there is no need to be careful about the order in which models are loaded since they can be placed deterministically regardless of the load order.
- **Implicit Placement**: Similarly, explicit placement means there is no chance that a model will end up being allocated to an incorrect NeuronCore Group.

Example: Automatic Multicore

Using explicit core placement it is possible to replicate a model to multiple NeuronCores simultaneously. This means that a single model object within python can utilize all available NeuronCores (or NeuronCores allocated to the process).

Environment Setup:

```
export NEURON_RT_NUM_CORES = '8'
```

Python Script:

```
import torch
import torch_neuron

with torch_neuron.experimental.multicore_context():
    m0 = torch.jit.load('model-with-1-neuron-pipeline-cores.pt')  # Loads replications␣
↪to nc0-nc7
```

(continues on next page)

(continued from previous page)

This addresses the last NEURONCORE_GROUP_SIZES issue of:

- **Manual Replication**: Since models can be automatically replicated to multiple NeuronCores, this means that applications no longer need to implement routing logic and perform multiple loads.

This API has a secondary benefit that the exact same loading logic can be used on an `inf1.xlarge` or an `inf1.6xlarge`. In either case, it will use all of the NeuronCores that are visible to the process. This means that no special logic needs to be coded for different instance types.

Example: Explicit Replication

Replication is also possible with the `neuron_cores_context()` API. The number of replications is chosen by `replications = floor(nc_count / cores_per_model)`.

Environment Setup:

```
export NEURON_RT_NUM_CORES = '8'
```

Python Script:

```
import torch
import torch_neuron

with torch_neuron.experimental.neuron_cores_context(start_nc=2, nc_count=4):
    m0 = torch.jit.load('model-with-2-neuron-pipeline-cores.pt')  # Loads replications␣
↪to nc2-nc5
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.34 PyTorch Neuron (torch-neuron) Supported operators

Current operator lists may be generated with these commands inside python:

```
import torch.neuron
print(*torch.neuron.get_supported_operations(), sep='\n')
```

8.34.1 PyTorch Neuron release [2.5.0.0]

Date: 11/23/2022

Added support for new operators:

- `aten::threshold`
- `aten::roll`
- `aten::instance_norm`
- `aten::amin`
- `aten::amax`

- aten::new_empty
- aten::new_ones
- aten::tril
- aten::triu
- aten::zero_
- aten::all
- aten::broadcast_tensors
- aten::broadcast_to
- aten::logical_and
- aten::logical_not
- aten::logical_or
- aten::logical_xor
- aten::_convolution_mode

Added **limited** support for new operators:

- **LSTM Operations. See:** *Developer Guide - PyTorch Neuron (torch-neuron) LSTM Support*
  - aten::lstm
  - aten::_pack_padded_sequence
  - aten::_pad_packed_sequence
- aten::norm: Supported when p argument is one of (1, 2, inf, -inf, 'fro')

8.34.2 PyTorch Neuron release [2.2.0.0]

Date: 03/25/2022

- Added full support for aten::max_pool2d_with_indices - (Was previously supported only when indices were unused).

8.34.3 PyTorch Neuron release [2.1.7.0]

Date: 01/20/2022

- Added support for new operators:
  - aten::bucketize
  - aten::any
  - aten::remainder
  - aten::clip
  - aten::repeat_interleave
  - aten::tensor_split
  - aten::split_with_sizes
  - aten::isnan

- aten::embedding_renorm_
- aten::dot
- aten::mv
- aten::hardsigmoid
- aten::hardswish
- aten::trunc
- aten::one_hot: Supported when num_classes is known at trace time. The dynamic version of this operation when num_classes = -1 is not supported.
- aten::adaptive_max_pool1d
- aten::adaptive_max_pool2d

PyTorch Neuron Release [2.0.536.0]

- The following are operators with limited support on Neuron. Unlike fully supported operators, these operators are not returned when using torch_neuron.get_supported_operations(). See each operator description for conditional support:
  - aten::max_pool2d_with_indices - Supported when indices outputs are not used by a downstream operation. This allows the operation to be compiled to Neuron when it is equivalent to an aten::max_pool2d.
  - aten::max_pool3d_with_indices - Supported when indices outputs are not used by a downstream operation. This allows the operation to be compiled to Neuron when it is equivalent to an aten::max_pool3d.
  - aten::where - Supported when used as a conditional selection (3-argument variant). Unsupported when used to generate a dynamic list of indices (1-argument variant). See torch.where().

PyTorch Neuron Release [2.0.318.0]

- Added support for new operators:
  - aten::empty_like
  - aten::log
  - aten::type_as
  - aten::movedim
  - aten::einsum
  - aten::argmax
  - aten::min
  - aten::argmin
  - aten::abs
  - aten::cos
  - aten::sin
  - aten::linear
  - aten::pixel_shuffle

- aten::group_norm
- aten::_weight_norm

PyTorch Neuron Release [1.5.21.0]

No change

PyTorch Neuron Release [1.5.7.0]

Added:

```
aten::erf
prim::DictConstruct
```

PyTorch Neuron Release [1.4.1.0]

No change

PyTorch Neuron Release [1.3.5.0]

Added:

```
aten::numel
aten::ones_like
aten::reciprocal
aten::topk
```

PyTorch Neuron Release [1.2.16.0]

No change

PyTorch Neuron Release [1.2.15.0]

No change

PyTorch Neuron Release [1.2.3.0]

Added:

```
aten::silu
aten::zeros_like
```

PyTorch Neuron Release [1.1.7.0]

Added:

```
aten::_shape_as_tensor
aten::chunk
aten::empty
aten::masked_fill
```

PyTorch Neuron Release [1.0.24045.0]

Added:

```
aten::__and__
aten::bmm
aten::clone
aten::expand_as
aten::fill_
aten::floor_divide
aten::full
aten::hardtanh
aten::hardtanh_
aten::le
aten::leaky_relu
aten::lt
aten::mean
aten::ne
aten::softplus
aten::unbind
aten::upsample_bilinear2d
```

PyTorch Neuron Release [1.0.1720.00]

Added:

```
aten::constant_pad_nd
aten::meshgrid
```

PyTorch Neuron Release [1.0.1532.0]

Added:

```
aten::ones
```

PyTorch Neuron Release [1.0.1522.0]

- No change

PyTorch Neuron Release [1.0.1386.0]

Added the following instructions. Please note, primitives are included in this list from this release.

```
aten::ceil
aten::clamp
aten::eq
aten::exp
aten::expand_as
aten::flip
aten::full_like
aten::ge
aten::gt
aten::log2
aten::log_softmax
aten::max
aten::neg
aten::relu
aten::rsqrt
aten::scalarImplicit
aten::sqrt
aten::squeeze
aten::stack
aten::sub
aten::sum
aten::true_divide
aten::upsample_nearest2d
prim::Constant
prim::GetAttr
prim::ImplicitTensorToNum
prim::ListConstruct
prim::ListUnpack
prim::NumToTensor
prim::TupleConstruct
prim::TupleUnpack
```

PyTorch Neuron Release [1.0.1168.0]

Added `aten::ScalarImplicit`

PyTorch Neuron Release [1.0.1001.0]

Added

```
aten::detach
aten::floor
aten::gelu
aten::pow
aten::sigmoid
aten::split
```

Removed (Reasons given alongside)

```
aten::embedding (does not meet performance criteria)
aten::erf (error function does not meet accuracy criteria)
aten::tf_dtype_from_torch (internal support function, not an operator)
```

PyTorch Neuron Release [1.0.825.0]

PyTorch Neuron Release [1.0.763.0]

- Please note. Starting with this release we will not publish primitives (prim::).
- Previous release inaccurately listed these operators as aten ops, they are not.

```
aten::tf_broadcastable_slice
aten::tf_padding
```

The following new operators are added in this release.

```
aten::Int
aten::arange
aten::contiguous
aten::div
aten::embedding
aten::erf
aten::expand
aten::eye
aten::index_select
aten::layer_norm
aten::matmul
aten::mm
aten::permute
aten::reshape
aten::rsub
aten::select
aten::size
aten::slice
```

(continues on next page)

(continued from previous page)

```
aten::softmax
aten::tf_dtype_from_torch
aten::to
aten::transpose
aten::unsqueeze
aten::view
aten::zeros
```

These operators were already supported previously (removing the two that were included by mistake)

```
aten::_convolution
aten::adaptive_avg_pool2d
aten::add
aten::add_
aten::addmm
aten::avg_pool2d
aten::batch_norm
aten::cat
aten::dimension_value
aten::dropout
aten::flatten
aten::max_pool2d
aten::mul
aten::relu_
aten::t
aten::tanh
aten::values
prim::Constant
prim::GetAttr
prim::ListConstruct
prim::ListUnpack
prim::TupleConstruct
prim::TupleUnpack
```

PyTorch Neuron Release [1.0.672.0]

No change

PyTorch Neuron Release [1.0.552.0]

```
aten::_convolution
aten::adaptive_avg_pool2d
aten::add
aten::add_
aten::addmm
aten::avg_pool2d
aten::batch_norm
aten::cat
aten::dimension_value
aten::dropout
```

(continues on next page)

(continued from previous page)

```
aten::flatten
aten::max_pool2d
aten::mul
aten::relu_
aten::t
aten::tanh
aten::tf_broadcastable_slice
aten::tf_padding
aten::values
prim::Constant
prim::GetAttr
prim::ListConstruct
prim::ListUnpack
prim::TupleConstruct
prim::TupleUnpack
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.35 Troubleshooting Guide for PyTorch Neuron (`torch-neuron`)

8.35.1 General Torch-Neuron issues

If you see an error about "Unknown builtin op: neuron::forward_1" like below, please ensure that import line "import torch_neuron" (to register the Neuron custom operation) is in the inference script before using torch.jit.load.

```
Unknown builtin op: neuron::forward_1.
Could not find any similar ops to neuron::forward_1. This op may not exist or may not be␣
↪currently supported in TorchScript.
```

8.35.2 TorchVision related issues

If you encounter an error like below, it is because latest torchvision version >= 0.7 is not compatible with Torch-Neuron 1.5.1. Please downgrade torchvision to version 0.6.1:

```
E   AttributeError: module 'torch.jit' has no attribute '_script_if_tracing'
```

8.35.3 torch.jit.trace issues

The /neuron-guide/neuron-frameworks/pytorch-neuron/api-compilation-python-api.rst uses the PyTorch `torch.jit.trace()` function to generate `ScriptModule` models for execution on Inferentia. Due to that, to execute your PyTorch model on Inferentia it must be torch-jit-traceable, otherwise you need to make sure your model is torch-jit-traceable. You can try modifying your underlying PyTorch model code to make it traceable. If it's not possible to change your model code, you can *write a wrapper around your model* that makes it torch-jit-traceable to compile it for Inferentia.

Please visit `torch.jit.trace()` to review the properties that a model must have to be torch-jit-traceable. The PyTorch-Neuron trace API *`torch_neuron.trace()`* accepts `**kwargs` for `torch.jit.trace()`. For example, you can use the `strict=False` flag to *compile models with dictionary outputs*.

Compiling models with outputs that are not torch-jit-traceable

To enable compilation of models with non torch-jit-traceable outputs, you can use a technique that involves writing a wrapper that converts the model's output into a form that is torch-jit-traceable. You can then compile the wrapped model for Inferentia using `torch_neuron.trace()`.

The following example uses a wrapper to compile a model with non torch-jit-traceable outputs. This model cannot be compiled for Inferentia in its current form because it outputs a list of tuples and tensors, which is not torch-jit-traceable.

```
import torch
import torch_neuron
import torch.nn as nn

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv = nn.Conv2d(1, 1, 3)

    def forward(self, x):
        a = self.conv(x) + 1
        b = self.conv(x) + 2
        c = self.conv(x) + 3
        # An output that is a list of tuples and tensors is not torch-traceable
        return [(a, b), c]

model = Model()
model.eval()

inputs = torch.rand(1, 1, 3, 3)

# Try to compile the model
model_neuron = torch.neuron.trace(model, inputs) # ERROR: This cannot be traced, we must␣
↪change the output format
```

To compile this model for Inferentia, we can write a wrapper around the model to convert its outputs into a tuple of tensors, which is torch-jit-traceable.

```
class NeuronCompatibilityWrapper(nn.Module):
    def __init__(self):
        super(NeuronCompatibilityWrapper, self).__init__()
        self.model = Model()

    def forward(self, x):
        out = self.model(x)
        # An output that is a tuple of tuples and tensors is torch-jit-traceable
        return tuple(out)
```

Now, we can successfully compile the model for Inferentia using the `NeuronCompatibilityWrapper` wrapper as follows:

```
model = NeuronCompatibilityWrapper()
model.eval()

# Compile the traceable wrapped model
model_neuron = torch.neuron.trace(model, inputs)
```

If the model's outputs must be in the original form, a second wrapper can be used to transform the outputs after compilation for Inferentia. The following example uses the `OutputFormatWrapper` wrapper to convert the compiled model's output back into the original form of a list of tuples and tensors.

```
class OutputFormatWrapper(nn.Module):
    def __init__(self):
        super(OutputFormatWrapper, self).__init__()
        self.traceable_model = NeuronCompatibilityWrapper()

    def forward(self, x):
        out = self.traceable_model(x)
        # Return the output in the original format of Model()
        return list(out)

model = OutputFormatWrapper()
model.eval()

# Compile the traceable wrapped model
model.traceable_model = torch.neuron.trace(model.traceable_model, inputs)
```

Compiling a submodule in a model that is not torch-jit-traceable

The following example shows how to compile a submodule that is part of a non torch-jit-traceable model. In this example, the top-level model `Outer` uses a dynamic flag, which is not torch-jit-traceable. However, the submodule `Inner` is torch-jit-traceable and can be compiled for Inferentia.

```
import torch
import torch_neuron
import torch.nn as nn

class Inner(nn.Module) :
    def __init__(self):
        super().__init__()
        self.conv = nn.Conv2d(1, 1, 3)

    def forward(self, x):
        return self.conv(x) + 1


class Outer(nn.Module):
    def __init__(self):
        super().__init__()
        self.inner = Inner()

    def forward(self, x, add_offset: bool = False):
        base = self.inner(x)
        if add_offset:
            return base + 1
        return base

model = Outer()
inputs = torch.rand(1, 1, 3, 3)
```

(continues on next page)

(continued from previous page)

```
# Compile the traceable wrapped submodule
model.inner = torch.neuron.trace(model.inner, inputs)

# TorchScript the model for serialization
script = torch.jit.script(model)
torch.jit.save(script, 'model.pt')

loaded = torch.jit.load('model.pt')
```

Alternatively, for usage scenarios in which the model configuration is static during inference, the dynamic flags can be hardcoded in a wrapper to make the model torch-jit-traceable and enable compiling the entire model for Inferentia. In this example, we assume the `add_offset` flag is always `True` during inference, so we can hardcode this conditional path in the `Static` wrapper to remove the dynmaic behavior and compile the entire model for Inferentia.

```
class Static(nn.Module):
    def __init__(self):
        super().__init__()
        self.outer = Outer()

    def forward(self, x):
        # hardcode `add_offset=True`
        output = self.outer(x, add_offset=True)
        return output

model = Static()

# We can now compile the entire model because `add_offset=True` is hardcoded in the␣
↪Static wrapper
model_neuron = torch.neuron.trace(model, inputs)
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

8.36 PyTorch Neuron (`torch-neuron`) release notes

Table of contents

- *Known Issues and Limitations - Updated 11/23/2022*
- *PyTorch Neuron release [2.5.0.0]*
- *PyTorch Neuron release [2.3.0.0]*
- *PyTorch Neuron release [2.2.0.0]*
- *PyTorch Neuron release [2.1.7.0]*
- *PyTorch Neuron release [2.0.536.0]*
- *PyTorch Neuron release [2.0.468.0]*
- *PyTorch Neuron release [2.0.392.0]*

- *PyTorch Neuron release [2.0.318.0]*
- *[1.8.1.1.5.21.0]*
- *[1.8.1.1.5.7.0]*
- *[1.8.1.1.4.1.0]*
- *[1.7.1.1.3.5.0]*
- *[1.7.1.1.2.16.0]*
- *[1.7.1.1.2.15.0]*
- *[1.7.1.1.2.3.0]*
- *[1.1.7.0]*
- *[1.0.1978.0]*
- *[1.0.1721.0]*
- *[1.0.1532.0]*
- *[1.0.1522.0]*
- *[1.0.1386.0]*
- *[1.0.1168.0]*
- *[1.0.1001.0]*
- *[1.0.825.0]*
- *[1.0.763.0]*
- *[1.0.672.0]*
- *[1.0.627.0]*

This document lists the release notes for the Pytorch-Neuron package.

8.36.1 Known Issues and Limitations - Updated 11/23/2022

Min & Max Accuracy

The index outputs of the `aten::argmin`, `aten::argmax`, `aten::min`, and `aten::max` operator implementations are sensitive to precision. For models that contain these operators and have `float32` inputs, we recommend using the `--fp32-cast=matmult --fast-math no-fast-relayout` compiler option to avoid numerical imprecision issues. Additionally, the `aten::min` and `aten::max` operator implementations do not currently support `int64` inputs when `dim=0`. For more information on precision and performance-accuracy tuning, see *Mixed precision and performance-accuracy tuning (neuron-cc)*.

Python 3.5

If you attempt to import torch.neuron from Python 3.5 you will see this error in 1.1.7.0 - please use Python 3.6 or greater:

```
File "/tmp/install_test_env/lib/python3.5/site-packages/torch_neuron/__init__.py", line␣
↪29
  f'Invalid dependency version torch=={torch.__version__}. '
                                                         ^
SyntaxError: invalid syntax
```

- Torchvision has dropped support for Python 3.5
- HuggingFace transformers has dropped support for Python 3.5

Torchvision

When versions of `torchvision` and `torch` are mismatched, this can result in exceptions when compiling `torchvision` based models. Specific versions of `torchvision` are built against each release of `torch`. For example:

- `torch==1.5.1` matches `torchvision==0.6.1`
- `torch==1.7.1` matches `torchvision==0.8.2`
- etc.

Simultaneously installing both `torch-neuron` and `torchvision` is the recommended method of correctly resolving versions.

Dynamic Batching

Dynamic batching does not work properly for some models that use the `aten::size` operator. When this issue occurs, the input batch sizes are not properly recorded at inference time, resulting in an error such as:

```
RuntimeError: The size of tensor a (X) must match the size of tensor b (Y) at non-
↪singleton dimension 0.
```

This error typically occurs when `aten::size` operators are partitioned to CPU. We are investigating a fix for this issue.

8.36.2 PyTorch Neuron release [2.5.0.0]

Date: 11/23/2022

New in this release

- Added PyTorch 1.12 support
- Added Python 3.8 support
- Added new operators support. See *PyTorch Neuron (torch-neuron) Supported operators*
- Added support for `aten::lstm`. See: *Developer Guide - PyTorch Neuron (torch-neuron) LSTM Support*
- **Improved logging:**
  - Improved error messages for specific compilation failure modes, including out-of-memory errors

  - Added a warning to show the code location of `prim::PythonOp` operations
  - Removed overly-verbose tracing messages
  - Added improved error messages for `neuron-cc` and `tensorflow` dependency issues
  - Added more debug information when an invalid dynamic batching configuration is used
- Added new experimental explicit NeuronCore placement API. See: *Experimental: NeuronCore Placement APIs*
- Added new guide for NeuronCore placement. See: *Developer Guide - PyTorch Neuron (torch-neuron) Core Placement*
- Improved `torch_neuron.trace()` performance when using large graphs
- Reduced host memory usage of loaded models in `libtorchneuron.so`
- Added `single_fusion_ratio_threshold` argument to `torch_neuron.trace()` to give more fine-grained control of partitioned graphs

Bug fixes

- Improved handling of tensor mutations which previously caused accuracy issues on certain models (i.e. yolor, yolov5)
- Fixed an issue where `inf` and `-inf` values would cause unexpected `NaN` values. This could occur with newer versions of `transformers`
- Fixed an issue where `torch.neuron.DataParallel()` would not fully utilize all NeuronCores for specific batch sizes
- **Fixed and improved operators:**
  - `aten::upsample_bilinear2d`: Improved error messages in cases where the operation cannot be supported
  - `aten::_convolution`: Added support for `output_padding` argument
  - `aten::div`: Added support for `rounding_mode` argument
  - `aten::sum`: Fixed to handle non-numeric data types
  - `aten::expand`: Fixed to handle scalar tensors
  - `aten::permute`: Fixed to handle negative indices
  - `aten::min`: Fixed to support more input types
  - `aten::max`: Fixed to support more input types
  - `aten::max_pool2d`: Fixed to support both 3-dimensional and 4-dimensional input tensors
  - `aten::Int`: Fixed an issue where long values would incorrectly lose precision
  - `aten::constant_pad_nd`: Fixed to correctly use non-0 padding values
  - `aten::pow`: Fixed to support more input types & values
  - `aten::avg_pool2d`: Added support for `count_include_pad` argument. Added support for `ceil_mode` argument if padding isn't specified
  - `aten::zero`: Fixed to handle scalars correctly
  - `prim::Constant`: Fixed an issue where `-inf` was incorrectly handled
  - Improved handling of scalars in arithmetic operators

8.36.3 PyTorch Neuron release [2.3.0.0]

Date: 04/29/2022

New in this release

- Added support PyTorch 1.11.
- Updated PyTorch 1.10 to version 1.10.2.
- End of support for torch-neuron 1.5, see eol-pt-15.
- Added support for new operators:
  - `aten::masked_fill_`
  - `aten::new_zeros`
  - `aten::frobenius_norm`

Bug fixes

- Improved `aten::gelu` accuracy
- Updated `aten::meshgrid` to support optional indexing argument introduced in `torch 1.10` , see PyTorch issue 50276

8.36.4 PyTorch Neuron release [2.2.0.0]

Date: 03/25/2022

New in this release

- Added full support for `aten::max_pool2d_with_indices` - (Was previously supported only when indices were unused).
- Added new torch-neuron packages compiled with `-D_GLIBCXX_USE_CXX11_ABI=1`, the new packages support PyTorch 1.8, PyTorch 1.9, and PyTorch 1.10. To install the additional packages compiled with `-D_GLIBCXX_USE_CXX11_ABI=1` please change the package repo index to `https://pip.repos.neuron.amazonaws.com (https://pip.repos.neuron.amazonaws.com/)/cxx11/`

8.36.5 PyTorch Neuron release [2.1.7.0]

Date: 01/20/2022

New in this release

- Added PyTorch 1.10 support
- Added new operators support, see *PyTorch Neuron (torch-neuron) Supported operators*
- Updated `aten::_convolution` to support 2d group convolution
- Updated `neuron::forward` operators to allocate less dynamic memory. This can increase performance on models with many input & output tensors.
- Updated `neuron::forward` to better handle batch sizes when `dynamic_batch_size=True`. This can increase performance at inference time when the input batch size is exactly equal to the traced model batch size.

Bug fixes

- Added the ability to `torch.jit.trace` a `torch.nn.Module` where a submodule has already been traced with `torch_neuron.trace()` on a CPU-type instance. Previously, if this had been executed on a CPU-type instance, an initialization exception would have been thrown.
- Fixed `aten::matmul` behavior on 1-dimensional by n-dimensional multiplies. Previously, this would cause a validation error.
- Fixed binary operator type promotion. Previously, in unusual situations, operators like `aten::mul` could produce incorrect results due to invalid casting.
- Fixed `aten::select` when index was -1. Previously, this would cause a validation error.
- Fixed `aten::adaptive_avg_pool2d` padding and striding behavior. Previously, this could generate incorrect results with specific configurations.
- Fixed an issue where dictionary inputs could be incorrectly traced when the tensor values had gradients.

8.36.6 PyTorch Neuron release [2.0.536.0]

Date: 01/05/2022

New in this release

- Added new operator support for specific variants of operations (See *PyTorch Neuron (torch-neuron) Supported operators*)
- Added optional `optimizations` keyword to `torch_neuron.trace()` which accepts a list of `Optimization` passes.

8.36.7 PyTorch Neuron release [2.0.468.0]

Date: 12/15/2021

New in this release

- Added support for `aten::cumsum` operation.
- Fixed `aten::expand` to correctly handle adding new dimensions.

8.36.8 PyTorch Neuron release [2.0.392.0]

Date: 11/05/2021

- Updated Neuron Runtime (which is integrated within this package) to `libnrt 2.2.18.0` to fix a container issue that was preventing the use of containers when /dev/neuron0 was not present. See details here neuron-runtime-release-notes.

8.36.9 PyTorch Neuron release [2.0.318.0]

Date: 10/27/2021

New in this release

- PyTorch Neuron 1.x now support Neuron Runtime 2.x (`libnrt.so` shared library) only.

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.
  - Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

- Introducing PyTorch 1.9.1 support (support for `torch==1.9.1`)
- Added `torch_neuron.DataParallel`, see ResNet-50 tutorial *[html]* and *Data Parallel Inference on Torch Neuron* application note.
- Added support for tracing on GPUs
- Added support for `ConvTranspose1d`
- Added support for new operators:
  - `aten::empty_like`
  - `aten::log`
  - `aten::type_as`
  - `aten::movedim`
  - `aten::einsum`
  - `aten::argmax`
  - `aten::min`
  - `aten::argmin`

  - aten::abs
  - aten::cos
  - aten::sin
  - aten::linear
  - aten::pixel_shuffle
  - aten::group_norm
  - aten::_weight_norm
- Added torch_neuron.is_available()

Resolved Issues

- Fixed a performance issue when using both the dynamic_batch_size=True trace option and --neuron-core-pipeline compiler option. Dynamic batching now uses OpenMP to execute pipeline batches concurrently.
- Fixed torch_neuron.trace issues:
  - Fixed a failure when the same submodule was traced with multiple inputs
  - Fixed a failure where some operations would fail to be called with the correct arguments
  - Fixed a failure where custom operators (torch plugins) would cause a trace failure
- Fixed variants of aten::upsample_bilinear2d when scale_factor=1
- Fixed variants of aten::expand using dim=-1
- Fixed variants of aten::stack using multiple different input data types
- Fixed variants of aten::max using indices outputs

8.36.10 [1.8.1.1.5.21.0]

Date: 08/12/2021

Summary

- Minor updates.

8.36.11 [1.8.1.1.5.7.0]

Date: 07/02/2021

Summary

- Added support for dictionary outputs using `strict=False` flag. See /neuron-guide/neuron-frameworks/pytorch-neuron/troubleshooting-guide.rst.
- Updated `aten::batch_norm` to correctly implement the `affine` flag.
- Added support for `aten::erf` and `prim::DictConstruct`. See *PyTorch Neuron (torch-neuron) Supported operators*.
- Added dynamic batch support. See /neuron-guide/neuron-frameworks/pytorch-neuron/api-compilation-python-api.rst.

8.36.12 [1.8.1.1.4.1.0]

Date: 5/28/2021

Summary

- Added support for PyTorch 1.8.1
  - Models compatibility
    - Models compiled with previous versions of PyTorch Neuron (<1.8.1) are compatible with PyTorch Neuron 1.8.1.
    - Models compiled with PyTorch Neuron 1.8.1 are not backward compatible with previous versions of PyTorch Neuron (<1.8.1) .
  - Updated tutorials to use Hugging Face Transformers 4.6.0.
  - Added a new set of forward operators (forward_v2)
  - Host memory allocation when loading the same model on multiple NeuronCores is significantly reduced
  - Fixed an issue where models would not deallocate all memory within a python session after being garbage collected.
  - Fixed a TorchScript/C++ issue where loading the same model multiple times would not use multiple NeuronCores by default.
- Fixed logging to no longer configure the root logger.
- Removed informative messages that were produced during compilations as warnings. The number of warnings reduced significantly.
- Convolution operator support has been extended to include ConvTranspose2d variants.
- Reduce the amount of host memory usage during inference.

8.36.13 [1.7.1.1.3.5.0]

Date: 4/30/2021

Summary

- ResNext models now functional with new operator support
- Yolov5 support refer to https://github.com/aws/aws-neuron-sdk/issues/253 note https://github.com/ultralytics/yolov5/pull/2953 which optimized YoloV5 for AWS Neuron
- Convolution operator support has been extended to include most Conv1d and Conv3d variants
- New operator support. Please see *PyTorch Neuron (torch-neuron) Supported operators* for the complete list of operators.

8.36.14 [1.7.1.1.2.16.0]

Date: 3/4/2021

Summary

- Minor enhancements.

8.36.15 [1.7.1.1.2.15.0]

Date: 2/24/2021

Summary

- Fix for CVE-2021-3177.

8.36.16 [1.7.1.1.2.3.0]

Date: 1/30/2021

Summary

- Made changes to allow models with -inf scalar constants to correctly compile
- Added new operator support. Please see *PyTorch Neuron (torch-neuron) Supported operators* for the complete list of operators.

8.36.17 [1.1.7.0]

Date: 12/23/2020

Summary

- We are dropping support for Python 3.5 in this release
- torch.neuron.trace behavior will now throw a RuntimeError in the case that no operators are compiled for neuron hardware
- torch.neuron.trace will now display compilation progress indicators (dots) as default behavior (neuron-cc must updated to the December release to greater to see this feature)
- Added new operator support. Please see *PyTorch Neuron (torch-neuron) Supported operators* for the complete list of operators.
- Extended the BERT pretrained tutorial to demonstrate execution on multiple cores and batch modification, updated the tutorial to accomodate changes in the Hugging Face Transformers code for version 4.0
- Added a tutorial for torch-serve which extends the BERT tutorial
- Added support for PyTorch 1.7

8.36.18 [1.0.1978.0]

Date: 11/17/2020

Summary

- Fixed bugs in comparison operators, and added remaining variantes (eq, ne, gt, ge, lt, le)
- Added support for prim::PythonOp - note that this must be run on CPU and not Neuron. We recommend you replace this code with PyTorch operators if possible
- Support for a series of new operators. Please see *PyTorch Neuron (torch-neuron) Supported operators* for the complete list of operators.
- Performance improvements to the runtime library
- Correction of a runtime library bug which caused models with large tensors to generate incorrect results in some cases

8.36.19 [1.0.1721.0]

Date: 09/22/2020

Summary

- Various minor improvements to the Pytorch autopartitioner feature
- Support for the operators aten::constant_pad_nd, aten::meshgrid
- Improved performance on various torchvision models. Of note are resnet50 and vgg16

8.36.20 [1.0.1532.0]

Date: 08/08/2020

Summary

- Various minor improvements to the Pytorch autopartitioner feature
- Support for the aten:ones operator

8.36.21 [1.0.1522.0]

Date: 08/05/2020

Summary

Various minor improvements.

8.36.22 [1.0.1386.0]

Date: 07/16/2020

Summary

This release adds auto-partitioning, model analysis and PyTorch 1.5.1 support, along with a number of new operators

Major New Features

- Support for Pytorch 1.5.1
- Introduce an automated operator device placement mechanism in torch.neuron.trace to run sub-graphs that contain operators that are not supported by the neuron compiler in native PyTorch. This new mechanism is on by default and can be turned off by adding argument fallback=False to the compiler arguments.
- Model analysis to find supported and unsupported operators in a model

Resolved Issues

8.36.23 [1.0.1168.0]

Date 6/11/2020

Summary

Major New Features

Resolved Issues

Known Issues and Limitations

8.36.24 [1.0.1001.0]

Date: 5/11/2020

Summary

Additional PyTorch operator support and improved support for model saving and reloading.

Major New Features

- Added Neuron Compiler support for a number of previously unsupported PyTorch operators. Please see :ref:`neuron-cc-ops-pytorch`for the complete list of operators.
- Add support for torch.neuron.trace on models which have previously been saved using torch.jit.save and then reloaded.

Resolved Issues

Known Issues and Limitations

8.36.25 [1.0.825.0]

Date: 3/26/2020

Summary

Major New Features

Resolved Issues

Known Issues and limitations

8.36.26 [1.0.763.0]

Date: 2/27/2020

Summary

Added Neuron Compiler support for a number of previously unsupported PyTorch operators. Please see *PyTorch Neuron (torch-neuron) Supported operators* for the complete list of operators.

Major new features

- None

Resolved issues

- None

8.36.27 [1.0.672.0]

Date: 1/27/2020

Summary

Major new features

Resolved issues

- Python 3.5 and Python 3.7 are now supported.

Known issues and limitations

Other Notes

8.36.28 [1.0.627.0]

Date: 12/20/2019

Summary

This is the initial release of torch-neuron. It is not distributed on the DLAMI yet and needs to be installed from the neuron pip repository.

Note that we are currently using a TensorFlow as an intermediate format to pass to our compiler. This does not affect any runtime execution from PyTorch to Neuron Runtime and Inferentia. This is why the neuron-cc installation must include [tensorflow] for PyTorch.

Major new features

Resolved issues

Known issues and limitations

Models TESTED

The following models have successfully run on neuron-inferentia systems

1. SqueezeNet
2. ResNet50
3. Wide ResNet50

Pytorch Serving

In this initial version there is no specific serving support. Inference works correctly through Python on Inf1 instances using the neuron runtime. Future releases will include support for production deployment and serving of models

Profiler support

Profiler support is not provided in this initial release and will be available in future releases

Automated partitioning

Automatic partitioning of graphs into supported and non-supported operations is not currently supported. A tutorial is available to provide guidance on how to manually parition a model graph. Please see pytorch-manual-partitioning-jn-tutorial

PyTorch dependency

Currently PyTorch support depends on a Neuron specific version of PyTorch v1.3.1. Future revisions will add support for 1.4 and future releases.

Trace behavior

In order to trace a model it must be in evaluation mode. For examples please see *ResNet50 model for Inferentia*

Six pip package is required

The Six package is required for the torch-neuron runtime, but it is not modeled in the package dependencies. This will be fixed in a future release.

Multiple NeuronCore support

If the num-neuroncores options is used the number of cores must be manually set in the calling shell environment variable for compilation and inference.

For example: Using the keyword argument compiler_args=['—num-neuroncores', '4'] in the trace call, requires NEURONCORE_GROUP_SIZES=4 to be set in the environment at compile time and runtime

CPU execution

At compilation time a constant output is generated for the purposes of tracing. Running inference on a non neuron instance will generate incorrect results. This must not be used. The following error message is generated to stderr:

```
Warning: Tensor output are ** NOT CALCULATED ** during CPU execution and only
indicate tensor shape
```

Other notes

- Python version(s) supported:
  - 3.6
- Linux distribution supported:
  - DLAMI Ubuntu 18 and Amazon Linux 2 (using Python 3.6 Conda environments)
  - Other AMIs based on Ubuntu 18
  - For Amazon Linux 2 please install Conda and use Python 3.6 Conda environment

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

NINE

TENSORFLOW NEURON

TensorFlow Neuron unlocks high-performance and cost-effective deep learning acceleration on AWS Trainium-based and Inferentia-based Amazon EC2 instances.

TensorFlow Neuron enables native TensorFlow models to be accelerated on Neuron devices, so you can use your existing framework application and get started easily with minimal code changes.

Inference

Setup Guide for Inf1

This document is relevant for: `Inf1`

9.1 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

9.1.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

TensorFlow 2.8.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx --allow-change-held-packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx -y
```

TensorFlow 2.7.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.7.0.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.7.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.7.0.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.7.0.2.5.6.0 -y
```

TensorFlow 2.6.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.6.3.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.6.3.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.6.3.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.6.3.2.5.6.0 -y
```

TensorFlow 2.5.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

TensorFlow 1.15.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

9.1.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.8.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"
```

TensorFlow 2.7.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.3.* "protobuf"
```

TensorFlow 2.6.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.5.* "protobuf"
```

TensorFlow 2.5.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

TensorFlow 1.15.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

9.1.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

TensorFlow 2.8.2

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx --allow-change-held-packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx -y
```

TensorFlow 2.7.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.7.0.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.7.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.7.0.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.7.0.2.5.6.0 -y
```

TensorFlow 2.6.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

#################################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.6.3.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.6.3.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.6.3.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.6.3.2.5.6.0 -y
```

TensorFlow 2.5.3

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

TensorFlow 1.15.5

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 -y
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum install tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.2 Update to latest TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

9.2.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum update tensorflow-model-server-neuronx -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx --allow-change-held-packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

###################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
###################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx -y
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################
#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum update tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum update tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Update Neuron TensorBoard
pip install --upgrade tensorboard-plugin-neuron

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

9.2.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc] "protobuf"
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron[cc]==2.5.3.* "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* neuron-cc "protobuf"
```

9.2.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum update tensorflow-model-server-neuronx -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx --allow-change-held-packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx -y
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.7.1 is not a supported framework
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
__main__ : error:   tensorflow-2.6.3 is not a supported framework
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
```

(continues on next page)

(continued from previous page)

```
sudo yum update tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=2.5.4.2.5.6.0 --allow-change-held-
packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==2.5.3.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx-2.5.4.2.5.6.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source tensorflow_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum update tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx=1.15.0.2.5.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron TensorFlow
pip install --upgrade tensorflow-neuron==1.15.5.* "protobuf"

# Optional: Update Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum update tensorflow-model-server-neuronx-1.15.0.2.5.6.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3 Install previous TensorFlow Neuron releases

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

This document is relevant for: `Inf1`

9.3.1 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.8.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.8.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.8.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.8.0.2.3.0.0 -y
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.7.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
```

(continues on next page)

(continued from previous page)

```
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.7.0.2.3.0.0 -y
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.6.3.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.6.3.2.3.0.0 -y
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.4.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.4.2.3.0.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.4.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.3.0.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.8.0.2.3.0.0 "protobuf"
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.3.0.0 "protobuf"
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.3.0.0 "protobuf"
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.3.0.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 neuron-cc==1.11.4.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.8.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.8.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.8.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.8.0.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.8.0.2.3.0.0 -y
```

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.7.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.7.0.2.3.0.0 -y
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.6.3.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.6.3.2.3.0.0 -y
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.4.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.4.2.3.0.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.3.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.3.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.3.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.3.0.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.3.0.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.2 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
```

(continues on next page)

(continued from previous page)

```
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.7.0.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.7.0.2.2.0.0 -y
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.6.3.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.6.3.2.2.0.0 -y
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.4.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.2.0.0 --allow-change-held-
↪packages -y
```

(continues on next page)

(continued from previous page)

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.4.2.2.0.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.3.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.2.0.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.7.1.2.2.0.0 "protobuf"
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.6.3.2.2.0.0 "protobuf"
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.3.2.2.0.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 neuron-cc==1.10.3.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.7.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.7.0.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.7.0.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.7.1.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.7.0.2.2.0.0 -y
```

TensorFlow 2.6.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.6.3.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.6.3.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.6.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.6.3.2.2.0.0 -y
```

TensorFlow 2.5.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.4.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.4.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.3.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.4.2.2.0.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.2.0.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.2.0.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.2.0.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.2.0.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.2.0.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.3 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
```

(continues on next page)

(continued from previous page)

```
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
```

(continues on next page)

(continued from previous page)

```
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##############################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 --allow-change-held-
↪packages -y
```

(continues on next page)

(continued from previous page)

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
```

(continues on next page)

(continued from previous page)

```
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.4 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.13.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.13.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
```

(continues on next page)

(continued from previous page)

```
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.13.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.13.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 --allow-change-held-
↪packages -y
```

(continues on next page)

(continued from previous page)

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.14.0 "protobuf"
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.1.14.0 "protobuf"
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.1.14.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.1.14.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.1.14.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 neuron-cc==1.9.1.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.14.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.1.14.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.1.14.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.1.14.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.1.14.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
```

(continues on next page)

(continued from previous page)

```
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.14.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.14.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.14.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.5 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
```

(continues on next page)

(continued from previous page)

```
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.6.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.4.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.4.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.4.0 --allow-change-held-
↪packages -y
```

(continues on next page)

(continued from previous page)

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.4.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.4.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.6.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.2.2.1.6.0 "protobuf"
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.4.0 "protobuf"
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.4.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.4.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.4.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 neuron-cc==1.9.1.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.3.2.1.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.2.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.3.2.1.6.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.4.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.4.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.4.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.4.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.4.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.4.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.4.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.4.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.6.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
```

(continues on next page)

(continued from previous page)

```
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.6.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.1.6.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.1.6.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.1.6.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.6 Install TensorFlow Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
```

(continues on next page)

(continued from previous page)

```
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.2.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.2.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
```

(continues on next page)

(continued from previous page)

```
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.2.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.2.2.0.5.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.5.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.5.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
```

(continues on next page)

(continued from previous page)

```
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.5.0 --allow-change-held-
↪packages -y
```

(continues on next page)

(continued from previous page)

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.5.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.5.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.2.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.0.5.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.1.2.0.5.0 "protobuf"
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.3.2.0.5.0 "protobuf"
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.4.2.0.5.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.2.0.5.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.2.0.5.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 neuron-cc==1.8.5.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.2.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.2.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.2.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.1.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.2.2.0.5.0 -y
```

TensorFlow 2.4.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.3.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.3.2.0.5.0 -y
```

TensorFlow 2.3.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.4.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.4.2.0.5.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.3.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.3.2.0.5.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
```

(continues on next page)

(continued from previous page)

```
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.2.0.5.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.0.5.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
```

(continues on next page)

(continued from previous page)

```
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.2.0.5.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.2.0.5.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.2.0.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.2.0.5.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.7 Install TensorFlow Neuron (Neuron 1.15.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
```

(continues on next page)

(continued from previous page)

```
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
```

(continues on next page)

(continued from previous page)

```
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.10.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.10.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.10.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.10.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
```

(continues on next page)

(continued from previous page)

```
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.10.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.10.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.10.0 "protobuf"
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.10.0 "protobuf"
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.10.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.10.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.10.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 neuron-cc==1.6.13.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.10.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.10.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.10.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.10.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.10.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.10.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.10.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.10.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.10.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.10.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.8 Install TensorFlow Neuron (Neuron 1.15.1)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.9 Install TensorFlow Neuron (Neuron 1.15.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
```

(continues on next page)

(continued from previous page)

```
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
```

(continues on next page)

(continued from previous page)

```
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.2.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

(continues on next page)

(continued from previous page)

```
# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.5.0.1.6.8.0 "protobuf"
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.4.2.1.6.8.0 "protobuf"
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
```

(continues on next page)

(continued from previous page)

```
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.3.3.1.6.8.0 "protobuf"
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.2.3.1.6.8.0 "protobuf"
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc]==2.1.4.1.6.8.0 "protobuf"
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 neuron-cc==1.6.13.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 2.5.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.5.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.5.0.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.5.1.1.6.8.0 -y
```

TensorFlow 2.4.2

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.4500.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.4.1.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.4.2.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.4.1.1.6.8.0 -y
```

TensorFlow 2.3.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.3.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.3.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.3.0.1.6.8.0 -y
```

TensorFlow 2.2.3

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.2.2.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.2.3.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.2.2.1.6.8.0 -y
```

TensorFlow 2.1.4

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
```

(continues on next page)

(continued from previous page)

```
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=2.1.4.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
```

(continues on next page)

(continued from previous page)

```
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==2.1.4.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-2.1.4.1.6.8.0 -y
```

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.6.8.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.6.8.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.6.8.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.3.10 Install TensorFlow Neuron (Neuron 1.14.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.0.0

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.5.1.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
```

(continues on next page)

(continued from previous page)

```
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.5.1.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.5.1.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron==2.1.0.0

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.5.1.0 -y
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 neuron-cc==1.5.5.0 "protobuf"
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

TensorFlow 1.15.5

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.5.1.0 -y
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install tensorflow-model-server-neuron-1.15.0.1.5.1.0 -y
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install tensorflow-model-server-neuron=1.15.0.1.5.1.0 --allow-change-held-
↪packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate TensorFlow
source activate aws_neuron_tensorflow_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron==1.15.5.1.5.1.0 "protobuf"

# Optional: Install Neuron TensorFlow model server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete tensorflow-model-server-neuron
sudo yum install tensorflow-model-server-neuron-1.15.0.1.5.1.0 -y
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

Tutorials (`tensorflow-neuron`)

Computer Vision Tutorials

- Tensorflow 1.x - OpenPose tutorial *[html]* [notebook]
- Tensorflow 1.x - ResNet-50 tutorial *[html]* [notebook]
- Tensorflow 1.x - YOLOv4 tutorial *[html]* [notebook]
- Tensorflow 1.x - YOLOv3 tutorial *[html]* [notebook]
- Tensorflow 1.x - SSD300 tutorial *[html]*
- Tensorflow 1.x - Keras ResNet-50 optimization tutorial *[html]* [notebook]

9.4 Running OpenPose on Inferentia

9.4.1 Note: this tutorial runs on tensorflow-neuron 1.x only

9.4.2 Introduction:

In this tutorial we will compile and deploy Openpose model for Inferentia. This jupyter notebook should run on an inf1.6xlarge instance for compilation and inference. The inference part of this tutorial requires inf1.6xlarge and not the compilation itself. For simplicity we will run this tutorial on a single instance but in real life scenario the compilation can be done on a compute c5.4xlarge instance and the deployment on the inf1 instance family.

In this tutorial we provide two main sections: 1. Compile the OpenPose model on inf1x6large. 2. Infer the same compiled model on inf1x6large.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the Tensorflow Installation Guide. You can select the Kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

9.4.3 Acknowledgement:

Many thanks to https://github.com/ildoonet for providing pretrained model as well as the image preprocessing/pose estimating infrastructure.

9.4.4 Download tensorflow pose net frozen graph.

[ ]:
```
!wget -c --tries=2 $( wget -q -O - http://www.mediafire.com/file/qlzzr20mpocnpa3/graph_
↪opt.pb | grep -o 'http*://download[^"]*' | tail -n 1 ) -O graph_opt.pb
```

9.4.5 Compile

Compile the pose net frozen graph into AWS Neuron compatible form. Network input image resolution is adjustable with argument –net_resolution (e. g., –net_resolution=656x368). The compiled model can accept arbitrary batch size input at runtime.

[ ]:
```
"""
Usage: python convert_graph_opt.py /path/to/graph_opt.pb /path/to/graph_opt_neuron.pb
"""
#import argparse
import numpy as np
import tensorflow as tf
from tensorflow.core.framework.tensor_shape_pb2 import TensorShapeProto
import tensorflow.neuron as tfn


def compile():
    #parser = argparse.ArgumentParser()
    #parser.add_argument('input_pb_path', help='Input serialized GraphDef protobuf')
    #parser.add_argument('output_pb_path', help='Ouput serialized GraphDef protobuf')
    #parser.add_argument('--net_resolution', default='656x368', help='Network resolution in␣
↪WxH format, e. g., --net_resolution=656x368')
    #parser.add_argument('--debug_verify', action='store_true')
    #args = parser.parse_args()

    input_pb_path = './graph_opt.pb'
    net_resolution = '656x368'
    output_pb_path = './graph_opt_neuron_' + net_resolution + '.pb'

    debug_verify = 'store_true'
    dim_w, dim_h = net_resolution.split('x')
    dim_w = int(dim_w)
    dim_h = int(dim_h)
    graph_def = tf.GraphDef()
    with open(input_pb_path, 'rb') as f:
        graph_def.ParseFromString(f.read())

    if debug_verify:
        np.random.seed(0)
        feed_dict = {'image:0': np.random.rand(1, dim_h, dim_w, 3)}
        output_name = 'Openpose/concat_stage7:0'
        with tf.Session(graph=tf.Graph()) as sess:
            tf.import_graph_def(graph_def, name='')
            result_reference = sess.run(output_name, feed_dict)

    preprocessing_ops = {'preprocess_divide', 'preprocess_divide/y', 'preprocess_subtract
↪', 'preprocess_subtract/y'}
    graph_def = nhwc_to_nchw(graph_def, preprocessing_ops)
    graph_def = inline_float32_to_float16(graph_def, preprocessing_ops)
    with tf.Session(graph=tf.Graph()) as sess:
        tf.import_graph_def(graph_def, name='')
        no_fuse_ops = preprocessing_ops.union({'Openpose/concat_stage7'})
        infer_graph = tfn.graph_util.inference_graph_from_session(
```

(continues on next page)

(continued from previous page)

```
            sess, shape_feed_dict={'image:0': [1, dim_h, dim_w, 3]}, output_tensors=[
↪'Openpose/concat_stage7:0'],
            no_fuse_ops=no_fuse_ops, dynamic_batch_size=True,
        )
    with open(output_pb_path, 'wb') as f:
        f.write(infer_graph.as_graph_def().SerializeToString())

    if debug_verify:
        with tf.Session(graph=infer_graph) as sess:
            result_compiled = sess.run(output_name, feed_dict)
        np.testing.assert_allclose(result_compiled, result_reference, rtol=1e-2, atol=1e-
↪3)


def inline_float32_to_float16(graph_def, preprocessing_ops):
    float32_enum = tf.float32.as_datatype_enum
    float16_enum = tf.float16.as_datatype_enum
    graph = tf.Graph()
    with graph.as_default():
        tf.import_graph_def(graph_def, name='')
    graph_def = graph.as_graph_def()
    for node in graph_def.node:
        if node.name in preprocessing_ops or node.op == 'Placeholder':
            cast_input_node_name = node.name
            continue
        if node.op == 'Const':
            if node.attr['dtype'].type == float32_enum:
                node.attr['dtype'].type = float16_enum
                tensor_def = node.attr['value'].tensor
                tensor_def.dtype = float16_enum
                if tensor_def.tensor_content:
                    const_np = np.frombuffer(tensor_def.tensor_content, dtype=np.
↪float32).astype(np.float16)
                    tensor_def.tensor_content = const_np.tobytes()
                elif len(tensor_def.float_val):
                    const_np = np.array(tensor_def.float_val).astype(np.float16).view(np.
↪uint16)
                    tensor_def.float_val[:] = []
                    tensor_def.half_val[:] = list(const_np)
                else:
                    raise NotImplementedError
        elif 'T' in node.attr and node.attr['T'].type == float32_enum:
            node.attr['T'].type = float16_enum
    for node in graph_def.node:
        if node.name == cast_input_node_name:
            node.name = '{}_PreCastFloat32ToFlot16'.format(node.name)
            input_node = node
            break
    cast_input_node = _gen_cast_node_def(cast_input_node_name, tf.float16, input_node)

    output_node = graph_def.node[-1]
    cast_output_node_name = output_node.name
```

(continues on next page)

(continued from previous page)

```
    output_node.name = '{}_PreCastFloat16ToFlot32'.format(output_node.name)
    cast_output_node = _gen_cast_node_def(cast_output_node_name, tf.float32, output_node)

    preprocessing_ops.add(input_node.name)
    new_graph_def = tf.GraphDef()
    new_graph_def.node.extend(graph_def.node)
    new_graph_def.node.append(cast_input_node)
    new_graph_def.node.append(cast_output_node)
    graph = tf.Graph()
    with graph.as_default():
        tf.import_graph_def(new_graph_def, name='')
    return graph.as_graph_def()


def nhwc_to_nchw(graph_def, preprocessing_ops):
    graph = tf.Graph()
    with graph.as_default():
        tf.import_graph_def(graph_def, name='')
    graph_def = graph.as_graph_def()
    node_name_to_node = {node.name: node for node in graph_def.node}
    for node in graph_def.node:
        if node.name in preprocessing_ops or node.op == 'Placeholder':
            transpose_input_node_name = node.name
            continue
        if node.op == 'Conv2D':
            node.attr['data_format'].s = b'NCHW'
            strides = node.attr['strides'].list.i
            strides[:] = [strides[0], strides[3], strides[1], strides[2]]
        elif node.op == 'BiasAdd':
            if node.name != 'probs/BiasAdd':
                node.attr['data_format'].s = b'NCHW'
        elif node.op == 'MaxPool':
            node.attr['data_format'].s = b'NCHW'
            ksize = node.attr['ksize'].list.i
            ksize[:] = [ksize[0], ksize[3], ksize[1], ksize[2]]
            strides = node.attr['strides'].list.i
            strides[:] = [strides[0], strides[3], strides[1], strides[2]]
        elif node.op in {'Concat', 'ConcatV2'}:
            node_axes = node_name_to_node[node.input[-1]]
            node_axes.attr['value'].tensor.int_val[:] = [1]
    for node in graph_def.node:
        if node.name == transpose_input_node_name:
            node.name = '{}_PreTransposeNHWC2NCHW'.format(node.name)
            input_node = node
            break
    transpose_input_node, transpose_input_perm_node = _gen_transpose_def(transpose_input_
↪node_name, [0, 3, 1, 2], input_node)

    output_node = graph_def.node[-1]
    transpose_output_node_name = output_node.name
    output_node.name = '{}_PreTransposeNCHW2NHWC'.format(output_node.name)
    transpose_output_node, transpose_output_perm_node = _gen_transpose_def(transpose_
↪output_node_name, [0, 2, 3, 1], output_node)
```

(continues on next page)

(continued from previous page)

```
    preprocessing_ops.add(input_node.name)
    preprocessing_ops.add(transpose_input_perm_node.name)
    new_graph_def = tf.GraphDef()
    new_graph_def.node.extend(graph_def.node)
    new_graph_def.node.append(transpose_input_perm_node)
    new_graph_def.node.append(transpose_input_node)
    new_graph_def.node.append(transpose_output_perm_node)
    new_graph_def.node.append(transpose_output_node)
    graph = tf.Graph()
    with graph.as_default():
        tf.import_graph_def(new_graph_def, name='')
    return graph.as_graph_def()


def _gen_cast_node_def(name, target_dtype, input_node):
    cast_node = tf.NodeDef(name=name, op='Cast')
    cast_node.input.append(input_node.name)
    cast_node.attr['DstT'].type = target_dtype.as_datatype_enum
    cast_node.attr['SrcT'].type = input_node.attr['T'].type
    cast_node.attr['Truncate'].b = False
    return cast_node


def _gen_transpose_def(name, perm, input_node):
    perm_node = tf.NodeDef(name='{}/perm'.format(name), op='Const')
    perm_node.attr['dtype'].type = tf.int32.as_datatype_enum
    tensor_def = perm_node.attr['value'].tensor
    tensor_def.dtype = tf.int32.as_datatype_enum
    tensor_def.tensor_shape.dim.append(TensorShapeProto.Dim(size=4))
    tensor_def.tensor_content = np.array(perm, dtype=np.int32).tobytes()
    transpose_node = tf.NodeDef(name=name, op='Transpose')
    transpose_node.input.append(input_node.name)
    transpose_node.input.append(perm_node.name)
    transpose_node.attr['T'].type = input_node.attr['T'].type
    transpose_node.attr['Tperm'].type = tf.int32.as_datatype_enum
    return transpose_node, perm_node
```

[ ]:
```
compile()

# Sample output will look like below:
# WARNING:tensorflow:From <ipython-input-3-27d3844cd753>:47: inference_graph_from_
↪session (from tensorflow_neuron.python.graph_util) is deprecated and will be removed␣
↪in a future version.
# Instructions for updating:
# Please refer to AWS documentation on Neuron integrated TensorFlow 2.0.
# INFO:tensorflow:Froze 0 variables.
# INFO:tensorflow:Converted 0 variables to const ops.
# INFO:tensorflow:fusing subgraph {subgraph neuron_op_ed41d2deb8c54255 with input␣
↪tensors ["<tf.Tensor 'preprocess_subtract0/_0:0' shape=(1, 3, 368, 656) dtype=float16>
↪"], output tensors ["<tf.Tensor 'Openpose/concat_stage7_PreCastFloat16ToFlot32:0'␣
↪shape=(1, 46, 82, 57) dtype=float16>"]} with neuron-cc
```

(continues on next page)

(continued from previous page)

```
# INFO:tensorflow:Number of operations in TensorFlow session: 474
# INFO:tensorflow:Number of operations after tf.neuron optimizations: 474
# INFO:tensorflow:Number of operations placed on Neuron runtime: 465
```

9.4.6 Deploy

Using same instance to deploy the model. In case of different deployment instance, launch a deployment inf1 instance and copy the AWS Neuron optimized tensorflow frozen graph graph_opt_neuron_656x368.pb to the deployment inf1 instance. The smallest instance type inf1.xlarge is sufficient for this demo.

Your graph_opt_neuron_656x368.pb can now be plugged into https://github.com/ildoonet seemlessly if you have tensorflow-neuron installed. When it is used at runtime, please ensure that the image resolution is the same as compile-time image resolution, i. e., 656x368.

Measure performance on the compiled frozen graph using dummy inputs.

[ ]:
```
"""
Copyright (C) 2020, Amazon.com. All Rights Reserved
"""
import os
import atexit
import time
import math
import json
from collections import OrderedDict, Counter
from contextlib import contextmanager, ContextDecorator
from functools import wraps
from tensorflow.python.client import session
from tensorflow.python.platform import tf_logging as logging


class measure_performance(ContextDecorator):
    """Convenient tool for performance measurements.
    Can be apply on tensorflow session.run, tf-serving unary gRPC calls, or a given␣
↪custom function.
    Usage:
    To generate performance report for the entire Python or gRPC-client process, insert
    the following function call before running inferences:
    `tfn.measure_performance()`
    Then latency/throughput report will be generated when the process terminates.
    Alternatively, it is possible to use `tfn.measure_performance` programmatically
    as a context manager. Performance measurement will be done for all inferences
    happening under this context. Report will be displayed as INFO level log when exiting
    the context. It is also possible to obtain a JSON format report in Python.
    For example:
    ```
    with tfn.measure_performance() as perf:
        ... (run some inferences) ...
    report_json = perf.report()
    report_full_json = perf.report(verbosity=1)
    ```
    """
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, func=None, window_size=1):
        self.perf_tracker = PerformanceTracker(window_size)
        atexit.register(self.perf_tracker.report)
        self._original_run = session.Session.run
        self._original_grpc_call = None
        if callable(func):
            self.perf_tracker.register_func(self._track_performance(func))
        else:
            session.Session.run = self._track_performance(session.Session.run)
            try:
                import grpc
                from tensorflow_serving.apis import prediction_service_pb2_grpc
                dummy_stub = prediction_service_pb2_grpc.PredictionServiceStub(grpc.
↪insecure_channel(''))
                self._grpc_callable_type = type(dummy_stub.Predict)
                self._original_grpc_call = self._grpc_callable_type.__call__
            except ImportError:
                pass
            if callable(self._original_grpc_call):
                self._grpc_callable_type.__call__ = self._track_performance(
                    grpc._channel._UnaryUnaryMultiCallable.__call__
                )

    def __enter__(self):
        return self.perf_tracker

    def __exit__(self, *exc):
        atexit.unregister(self.perf_tracker.report)
        self.perf_tracker.report()
        session.Session.run = self._original_run
        if self._original_grpc_call is not None:
            self._grpc_callable_type.__call__ = self._original_grpc_call
        return False

    def _track_performance(self, func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            start = time.time()
            result = func(*args, **kwargs)
            end = time.time()
            self.perf_tracker.add_timestamps(start, end)
            return result
        return wrapper


class PerformanceTracker(ContextDecorator):

    description = (
        "Latency unit: second. Throughput unit: number of batched inferences per second.
↪"
        "Reported throughput is a lower bound of the actual throughput as inferences "
```

(continues on next page)

(continued from previous page)

```
        "spanning across window boundaries are not counted towards any of the windows. "
        "'Quiet' periods (i. e., window buckets where the inference function is not␣
↪called) "
        "are not counted towards the reported average throughput."
    )

    def __init__(self, window_size):
        self.window_size = window_size
        self.timestamps_list = []
        self._func = None

    def __call__(self, *args, **kwargs):
        return self._func(*args, **kwargs)

    def register_func(self, func):
        self._func = func

    def add_timestamps(self, start, end):
        self.timestamps_list.append([start, end])

    def report(self, verbosity=0):
        if self.timestamps_list:
            latency_list = [end - start for start, end in self.timestamps_list]
            latency_json = {
                'p50': percentile(latency_list, 50),
                'p90': percentile(latency_list, 90),
                'p99': percentile(latency_list, 99),
                'p100': percentile(latency_list, 100),
            }
            bucketed_timestamps = [self._get_bucket(start, end) for start, end in self.
↪timestamps_list]
            counted_buckets = Counter(item for item in bucketed_timestamps if item is␣
↪not None)
            bucket_throughputs = [(key, value / self.window_size) for key, value in␣
↪sorted(counted_buckets.items())]
            busy_throughputs = list(OrderedDict((key, value) for key, value in bucket_
↪throughputs).values())
            throughput_json = {
                'peak': max(busy_throughputs),
                'median': percentile(busy_throughputs, 50),
                'average': sum(busy_throughputs) / len(busy_throughputs),
            }
            if verbosity > 0:
                throughput_json['trend'] = busy_throughputs
            report_json = {
                'pid': os.getpid(),
                'throughput': throughput_json,
                'latency': latency_json,
                'description': PerformanceTracker.description,
            }
            with _logging_show_info():
                logging.info('performance report:\n{}'.format(json.dumps(report_json,␣
↪indent=4)))
```

(continues on next page)

(continued from previous page)

```
            return report_json

    def _get_bucket(self, start, end):
        bucketed_start = math.floor(start / self.window_size) * self.window_size
        bucketed_end = math.ceil(end / self.window_size) * self.window_size
        if bucketed_end - bucketed_start == self.window_size:
            return bucketed_start
        else:
            return None


def percentile(number_list, percent):
    pos_float = len(number_list) * percent / 100
    max_pos = len(number_list) - 1
    pos_floor = min(math.floor(pos_float), max_pos)
    pos_ceil = min(math.ceil(pos_float), max_pos)
    number_list = sorted(number_list)
    return number_list[pos_ceil] if pos_float - pos_floor > 0.5 else number_list[pos_
↪floor]


@contextmanager
def _logging_show_info():
    try:
        verbosity = logging.get_verbosity()
        logging.set_verbosity(logging.INFO)
        yield
    finally:
        logging.set_verbosity(verbosity)
```

[ ]:
```
"""
Below are the inputs for compiled frozen graph

pb_path is a /path/graph_opt_neuron_656x368.pb
num_thread = 8 ( Number of threads that work on each tensorflow session )
batch_size =1 ( batch_size )
net_resolution ,default=656x368
num_inferences = 200
"""
import os
from concurrent import futures
import numpy as np
import tensorflow as tf
import tensorflow.neuron as tfn

def run_with_dummy(sess, dummy_feed_dict, num_inferences):
    for _ in range(num_inferences):
        sess.run('Openpose/concat_stage7:0', dummy_feed_dict)

def main():
    NUM_NEURON_CORES = 16
    pb_path = './graph_opt_neuron_656x368.pb'
```

(continues on next page)

(continued from previous page)

```
    num_thread = 8
    batch_size = 1
    net_resolution = '656x368'
    num_inferences = 200
    dim_w, dim_h = net_resolution.split('x')
    dim_w = int(dim_w)
    dim_h = int(dim_h)
    graph_def = tf.GraphDef()
    with open(pb_path, 'rb') as f:
        graph_def.ParseFromString(f.read())

    graph_def = tfn.graph_util.tag_multicore(graph_def, NUM_NEURON_CORES)

    with tfn.measure_performance() as perf:
        with tf.Session(graph=tf.Graph()) as sess:
            tf.import_graph_def(graph_def, name='')
            input_name = 'image:0'
            input_shape = sess.graph.get_tensor_by_name(input_name).shape.as_list()
            input_shape[0] = batch_size
            input_shape[1] = dim_h
            input_shape[2] = dim_w
            dummy_feed_dict = {input_name: np.zeros(input_shape).astype(np.float32)}
            with futures.ThreadPoolExecutor(max_workers=num_thread) as executor:
                fut_list = [executor.submit(run_with_dummy, sess, dummy_feed_dict, num_
↪inferences) for _ in range(num_thread)]
                res_list = [fut.result() for fut in fut_list]

main()

# Sample output will look like below:
# INFO:tensorflow:performance report:
# {
#    "pid": 17713,
#    "throughput": {
#        "peak": 66.0,
#        "median": 64.0,
#        "average": 61.56521739130435
#    },
#    "latency": {
#        "p50": 0.1106414794921875,
#        "p90": 0.11212301254272461,
#        "p99": 0.11337876319885254,
#        "p100": 7.08282732963562
#    },
#    "description": "Latency unit: second. Throughput unit: number of batched inferences␣
↪per second. Reported throughput is a lower bound of the actual throughput as␣
↪inferences spanning across window boundaries are not counted towards any of the␣
↪windows. 'Quiet' periods (i. e., window buckets where the inference function is not␣
↪called) are not counted towards the reported average throughput."
# }
```

9.5 Running ResNet50 on Inferentia

9.5.1 Note: this tutorial runs on tensorflow-neuron 1.x only

9.5.2 Introduction:

In this tutorial we will compile and deploy ResNet50 model for Inferentia. In this tutorial we provide two main sections: 1. Compile the ResNet50 model. 2. Infer the same compiled model.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the Tensorflow Installation Guide. You can select the Kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

Instructions of how to setup Neuron Tensorflow environment and run the tutorial as a Jupyter notebook are available in the Tensorflow Quick Setup

9.5.3 Compile for Neuron

A trained model must be compiled to Inferentia target before it can be deployed on Inferentia instances. In this step we compile the Keras ResNet50 model and export it as a SavedModel which is an interchange format for TensorFlow models. At the end of compilation, the compiled SavedModel is saved in resnet50_neuron local directory:

[ ]:
```
import os
import time
import shutil
import tensorflow as tf
import tensorflow.neuron as tfn
import tensorflow.compat.v1.keras as keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

# Create a workspace
WORKSPACE = './ws_resnet50'
os.makedirs(WORKSPACE, exist_ok=True)

# Prepare export directory (old one removed)
model_dir = os.path.join(WORKSPACE, 'resnet50')
compiled_model_dir = os.path.join(WORKSPACE, 'resnet50_neuron')
shutil.rmtree(model_dir, ignore_errors=True)
shutil.rmtree(compiled_model_dir, ignore_errors=True)

# Instantiate Keras ResNet50 model
keras.backend.set_learning_phase(0)
keras.backend.set_image_data_format('channels_last')

model = ResNet50(weights='imagenet')

# Export SavedModel
tf.saved_model.simple_save(
    session            = keras.backend.get_session(),
    export_dir         = model_dir,
    inputs             = {'input': model.inputs[0]},
```

(continues on next page)

(continued from previous page)

```
    outputs             = {'output': model.outputs[0]})

# Compile using Neuron
tfn.saved_model.compile(model_dir, compiled_model_dir)
```

[ ]:
```
!ls
```

9.5.4 Deploy on Inferentia

Using same instance to deploy the model. In case of different deployment instance, launch a deployment inf1 instance and copy compiled model to the deployment inf1 instance.

Download the example image, and install pillow module for inference on deployement instance

[ ]:
```
!curl -O https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/
↪kitten_small.jpg
!pip install pillow  # Necessary for loading images
```

After downloading the example image, run the inference.

[ ]:
```
import os
import time
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import resnet50

tf.keras.backend.set_image_data_format('channels_last')

# Create input from image
img_sgl = image.load_img('kitten_small.jpg', target_size=(224, 224))
img_arr = image.img_to_array(img_sgl)
img_arr2 = np.expand_dims(img_arr, axis=0)
img_arr3 = resnet50.preprocess_input(img_arr2)

# Load model
COMPILED_MODEL_DIR = './ws_resnet50/resnet50_neuron/'
predictor_inferentia = tf.contrib.predictor.from_saved_model(COMPILED_MODEL_DIR)

# Run inference
model_feed_dict={'input': img_arr3}
infa_rslts = predictor_inferentia(model_feed_dict);

# Display results
print(resnet50.decode_predictions(infa_rslts["output"], top=5)[0])

# Sample output will look like below:
#[('n02123045', 'tabby', 0.68817204), ('n02127052', 'lynx', 0.12701613), ('n02123159', 'tiger_
↪cat', 0.08736559), ('n02124075', 'Egyptian_cat', 0.063844085), ('n02128757', 'snow_leopard',␣
↪0.009240591)]
```

(continues on next page)

(continued from previous page)

This document is relevant for: Inf1

9.6 Working with YOLO v4 using AWS Neuron SDK

The /src/examples/tensorflow/yolo_v4_demo/evaluate.ipynb notebook contains an example on how to take an open source YOLO v4 models, and run it on AWS Inferentia.

9.6.1 Optimizing image pre-processing and post-processing for object detection models

End-to-end object detection pipelines usually contain image pre-post-processing operators that cannot run efficiently on Inferentia. DecodeJPEG and NonMaxSuppression are typical examples. In practice, we may simply place these operators on CPU using the AWS Neuron machine learning framework integration. However, Inferentia is such a high performance machine learning accelerator that, once the model successfully compiles and runs, these simple pre-post-processing operators can become the new performance bottleneck! In this tutorial, we explain some commonly used tensorflow techniques for optimizing the performance of these pre-post-processing operators so that we can fully unleash the potential of Inferentia.

1. Write JPEG decoding and image shifting/scaling as tensorflow operators.

In `yolo_v4_coco_saved_model.py`, you may find the following code snippet.

```
import tensorflow as tf
...

def YOLOv4(...
    ...
    x, image_shape = layers.Lambda(lambda t: preprocessor(t, input_shape))(inputs)

    # cspdarknet53
    x = conv2d_unit(x, i32, 3, strides=1, padding='same')
...

def decode_jpeg_resize(input_tensor, image_size):
    tensor = tf.image.decode_png(input_tensor, channels=3)
    shape = tf.shape(tensor)
    tensor = tf.cast(tensor, tf.float32)
    tensor = tf.image.resize(tensor, image_size)
    tensor /= 255.0
    return tf.cast(tensor, tf.float16), shape

def preprocessor(input_tensor, image_size):
    with tf.name_scope('Preprocessor'):
        tensor = tf.map_fn(
            partial(decode_jpeg_resize, image_size=image_size), input_tensor,
            dtype=(tf.float16, tf.int32), back_prop=False, parallel_iterations=16)
    return tensor
```

Comparing with the implementation in the original repo, our difference is the use of `tf.image.decode_png` and `tf.image.resize`, along with a small number of scaling/casting operators. After this modification, the generated

tensorflow SavedModel now takes JPEG image raw bytes as input, instead of a float32 array representing the image. When the image resolution is 608x608, this technique effectively reduces the input image size from 4.4 MB to the size of a typical JPEG image, which can be as little as hundreds of KB. When the tensorflow SavedModel is deployed through tensorflow/serving, this technique can very effectively reduce the gRPC transfer overhead of input images.

2. Replace non-max suppression (NMS) operations by `tf.image.combined_non_max_suppression`.

Another difference of our implementation is the treatment of non-max suppression, a commmonly used operation for removing redundant bounding boxes that overlap with other boxes. In an object detection scenario represented by the COCO dataset where the number of output classes is large, the hand-fused `tf.image.combined_non_max_suppression <https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/image/combined_non_max_suppression>`__ operator can parallelize multi-class NMS on CPU in a very efficient manner. With proper use of this operator, the bounding box post-processing step has a less chance of becoming the performance bottleneck in the end-to-end object detection pipeline.

The following sample code (from `yolo_v4_coco_saved_model.py`) demonstrates our method of writing the bounding box post-processing step using efficient tensorflow operations.

```
...
    def filter_boxes(outputs):
        boxes_l, boxes_m, boxes_s, box_scores_l, box_scores_m, box_scores_s, image_shape␣
↪= outputs
        boxes_l, box_scores_l = filter_boxes_one_size(boxes_l, box_scores_l)
        boxes_m, box_scores_m = filter_boxes_one_size(boxes_m, box_scores_m)
        boxes_s, box_scores_s = filter_boxes_one_size(boxes_s, box_scores_s)
        boxes = tf.concat([boxes_l, boxes_m, boxes_s], axis=0)
        box_scores = tf.concat([box_scores_l, box_scores_m, box_scores_s], axis=0)
        image_shape_wh = image_shape[1::-1]
        image_shape_whwh = tf.concat([image_shape_wh, image_shape_wh], axis=-1)
        image_shape_whwh = tf.cast(image_shape_whwh, tf.float32)
        boxes *= image_shape_whwh
        boxes = tf.expand_dims(boxes, 0)
        box_scores = tf.expand_dims(box_scores, 0)
        boxes = tf.expand_dims(boxes, 2)
        nms_boxes, nms_scores, nms_classes, valid_detections = tf.image.combined_non_max_
↪suppression(
            boxes,
            box_scores,
            max_output_size_per_class=nms_top_k,
            max_total_size=nms_top_k,
            iou_threshold=nms_thresh,
            score_threshold=conf_thresh,
            pad_per_class=False,
            clip_boxes=False,
            name='CombinedNonMaxSuppression',
        )
        return nms_boxes[0], nms_scores[0], nms_classes[0]

    def filter_boxes_one_size(boxes, box_scores):
        box_class_scores = tf.reduce_max(box_scores, axis=-1)
        keep = box_class_scores > conf_thresh
        boxes = boxes[keep]
        box_scores = box_scores[keep]
        return boxes, box_scores
```

(continues on next page)

(continued from previous page)

```
    def batch_yolo_out(outputs):
        with tf.name_scope('yolo_out'):
            b_output_lr, b_output_mr, b_output_sr, b_image_shape = outputs
            with tf.name_scope('process_feats'):
                b_boxes_l, b_box_scores_l = batch_process_feats(b_output_lr, anchors,␣
↪masks[0])
            with tf.name_scope('process_feats'):
                b_boxes_m, b_box_scores_m = batch_process_feats(b_output_mr, anchors,␣
↪masks[1])
            with tf.name_scope('process_feats'):
                b_boxes_s, b_box_scores_s = batch_process_feats(b_output_sr, anchors,␣
↪masks[2])
            with tf.name_scope('filter_boxes'):
                b_nms_boxes, b_nms_scores, b_nms_classes = tf.map_fn(
                    filter_boxes, [b_boxes_l, b_boxes_m, b_boxes_s, b_box_scores_l, b_
↪box_scores_m, b_box_scores_s, b_image_shape],
                    dtype=(tf.float32, tf.float32, tf.float32), back_prop=False,␣
↪parallel_iterations=16)
        return b_nms_boxes, b_nms_scores, b_nms_classes

    boxes_scores_classes = layers.Lambda(batch_yolo_out)([output_lr, output_mr, output_
↪sr, image_shape])
...
```

For other advanced data input/output pipeline optimization techniques, please refer to https://www.tensorflow.org/guide/data#preprocessing_data.

This document is relevant for: `Inf1`

9.7 Evaluate YOLO v3 on Inferentia

9.7.1 Note: this tutorial runs on tensorflow-neuron 1.x only

9.7.2 Introduction

This tutorial walks through compiling and evaluating YOLO v3 model on Inferentia using the AWS Neuron SDK.

In this tutorial we provide two main sections:

1. Download Dataset and Generate Pretrained SavedModel
2. Compile the YOLO v3 model.
3. Deploy the same compiled model.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the Tensorflow Installation Guide. You can select the Kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

Instructions of how to setup Neuron Tensorflow environment and run the tutorial as a Jupyter notebook are available in the Tutorial main page Tensorflow-YOLO_v3 Tutorial

9.7.3 Prerequisites

This demo requires the following pip packages:

`pillow matplotlib pycocotools`

[ ]:
```
import sys
!{sys.executable} -m pip install pillow matplotlib pycocotools==2.0.2 --force --extra-
↪index-url=https://pip.repos.neuron.amazonaws.com
```

9.7.4 Part 1: Download Dataset and Generate Pretrained SavedModel

Download COCO 2017 validation dataset

We start by downloading the COCO validation dataset, which we will use to validate our model. The COCO 2017 dataset is widely used for object-detection, segmentation and image captioning.

[ ]:
```
!curl -LO http://images.cocodataset.org/zips/val2017.zip
!curl -LO http://images.cocodataset.org/annotations/annotations_trainval2017.zip
!unzip -q val2017.zip
!unzip annotations_trainval2017.zip
```

[ ]:
```
!ls
```

9.7.5 Generate YOLO v3 tensorflow SavedModel (pretrained on COCO 2017 dataset)

Script yolo_v3_coco_saved_model.py will generate a tensorflow SavedModel using pretrained weights from https://github.com/YunYang1994/tensorflow-yolov3/releases/download/v1.0/yolov3_coco.tar.gz.

[ ]:
```
%run yolo_v3_coco_saved_model.py ./yolo_v3_coco_saved_model
```

This tensorflow SavedModel can be loaded as a tensorflow predictor. When a JPEG format image is provided as input, the output result of the tensorflow predictor contains information for drawing bounding boxes and classification results.

[ ]:
```
import json
import tensorflow as tf
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# launch predictor and run inference on an arbitrary image in the validation dataset
yolo_pred_cpu = tf.contrib.predictor.from_saved_model('./yolo_v3_coco_saved_model')
image_path = './val2017/000000581781.jpg'
with open(image_path, 'rb') as f:
    feeds = {'image': [f.read()]}
results = yolo_pred_cpu(feeds)

# load annotations to decode classification result
with open('./annotations/instances_val2017.json') as f:
    annotate_json = json.load(f)
```

(continues on next page)

(continued from previous page)

```
label_info = {idx+1: cat['name'] for idx, cat in enumerate(annotate_json['categories'])}

# draw picture and bounding boxes
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(Image.open(image_path).convert('RGB'))
wanted = results['scores'][0] > 0.1
for xyxy, label_no_bg in zip(results['boxes'][0][wanted], results['classes'][0][wanted]):
    xywh = xyxy[0], xyxy[1], xyxy[2] - xyxy[0], xyxy[3] - xyxy[1]
    rect = patches.Rectangle((xywh[0], xywh[1]), xywh[2], xywh[3], linewidth=1,␣
↪edgecolor='g', facecolor='none')
    ax.add_patch(rect)
    rx, ry = rect.get_xy()
    rx = rx + rect.get_width() / 2.0
    ax.annotate(label_info[label_no_bg + 1], (rx, ry), color='w', backgroundcolor='g',␣
↪fontsize=10,
                ha='center', va='center', bbox=dict(boxstyle='square,pad=0.01', fc='g',␣
↪ec='none', alpha=0.5))
plt.show()
```

9.7.6 Part 2: Compile the Pretrained SavedModel for Neuron

We make use of the Python compilation API `tfn.saved_model.compile` that is available in `tensorflow-neuron<2`. For the purpose of reducing Neuron runtime overhead, it is necessary to make use of arguments `no_fuse_ops` and `minimum_segment_size`. Compiled model is saved in ./yolo_v3_coco_saved_model_neuron.

[ ]:
```
import shutil
import tensorflow as tf
import tensorflow.neuron as tfn


def no_fuse_condition(op):
    return op.name.startswith('Preprocessor') or op.name.startswith('Postprocessor')

with tf.Session(graph=tf.Graph()) as sess:
    tf.saved_model.loader.load(sess, ['serve'], './yolo_v3_coco_saved_model')
    no_fuse_ops = [op.name for op in sess.graph.get_operations() if no_fuse_
↪condition(op)]
shutil.rmtree('./yolo_v3_coco_saved_model_neuron', ignore_errors=True)
result = tfn.saved_model.compile(
    './yolo_v3_coco_saved_model', './yolo_v3_coco_saved_model_neuron',
    # to enforce trivial compilable subgraphs to run on CPU
    no_fuse_ops=no_fuse_ops,
    minimum_segment_size=100,
    batch_size=2,
    dynamic_batch_size=True,
)
print(result)
```

9.7.7 Deploy the model on Inferentia

9.7.8 Part 3:Evaluate Model Quality after Compilation

Define evaluation functions

We first define some handy helper functions for running evaluation on the COCO 2017 dataset.

[ ]:
```
import os
import json
import time
import numpy as np
import tensorflow as tf
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval


def cocoapi_eval(jsonfile,
                 style,
                 coco_gt=None,
                 anno_file=None,
                 max_dets=(100, 300, 1000)):
    """
    Args:
        jsonfile: Evaluation json file, eg: bbox.json, mask.json.
        style: COCOeval style, can be `bbox` , `segm` and `proposal`.
        coco_gt: Whether to load COCOAPI through anno_file,
                 eg: coco_gt = COCO(anno_file)
        anno_file: COCO annotations file.
        max_dets: COCO evaluation maxDets.
    """
    assert coco_gt is not None or anno_file is not None

    if coco_gt is None:
        coco_gt = COCO(anno_file)
    print("Start evaluate...")
    coco_dt = coco_gt.loadRes(jsonfile)
    if style == 'proposal':
        coco_eval = COCOeval(coco_gt, coco_dt, 'bbox')
        coco_eval.params.useCats = 0
        coco_eval.params.maxDets = list(max_dets)
    else:
        coco_eval = COCOeval(coco_gt, coco_dt, style)
    coco_eval.evaluate()
    coco_eval.accumulate()
    coco_eval.summarize()
    return coco_eval.stats


def bbox_eval(anno_file, bbox_list):
    coco_gt = COCO(anno_file)

    outfile = 'bbox_detections.json'
```

(continues on next page)

(continued from previous page)

```
    print('Generating json file...')
    with open(outfile, 'w') as f:
        json.dump(bbox_list, f)

    map_stats = cocoapi_eval(outfile, 'bbox', coco_gt=coco_gt)
    return map_stats


def get_image_as_bytes(images, eval_pre_path):
    batch_im_id_list = []
    batch_im_name_list = []
    batch_img_bytes_list = []
    n = len(images)
    batch_im_id = []
    batch_im_name = []
    batch_img_bytes = []
    for i, im in enumerate(images):
        im_id = im['id']
        file_name = im['file_name']
        if i % eval_batch_size == 0 and i != 0:
            batch_im_id_list.append(batch_im_id)
            batch_im_name_list.append(batch_im_name)
            batch_img_bytes_list.append(batch_img_bytes)
            batch_im_id = []
            batch_im_name = []
            batch_img_bytes = []
        batch_im_id.append(im_id)
        batch_im_name.append(file_name)

        with open(os.path.join(eval_pre_path, file_name), 'rb') as f:
            batch_img_bytes.append(f.read())
    return batch_im_id_list, batch_im_name_list, batch_img_bytes_list


def analyze_bbox(results, batch_im_id, _clsid2catid):
    bbox_list = []
    k = 0
    for boxes, scores, classes in zip(results['boxes'], results['scores'], results[
↪'classes']):
        if boxes is not None:
            im_id = batch_im_id[k]
            n = len(boxes)
            for p in range(n):
                clsid = classes[p]
                score = scores[p]
                xmin, ymin, xmax, ymax = boxes[p]
                catid = (_clsid2catid[int(clsid)])
                w = xmax - xmin + 1
                h = ymax - ymin + 1

                bbox = [xmin, ymin, w, h]
                # Round to the nearest 10th to avoid huge file sizes, as COCO suggests
```

(continues on next page)

(continued from previous page)

```
                bbox = [round(float(x) * 10) / 10 for x in bbox]
                bbox_res = {
                    'image_id': im_id,
                    'category_id': catid,
                    'bbox': bbox,
                    'score': float(score),
                }
                bbox_list.append(bbox_res)
        k += 1
    return bbox_list
```

Here is the actual evaluation loop. To fully utilize all four cores on one Inferentia, the optimal setup is to run multi-threaded inference using a `ThreadPoolExecutor`. The following cell is a multi-threaded adaptation of the evaluation routine at https://github.com/miemie2013/Keras-YOLOv4/blob/910c4c6f7265f5828fceed0f784496a0b46516bf/tools/cocotools.py#L97.

[ ]:
```
from concurrent import futures

def evaluate(yolo_predictor, images, eval_pre_path, anno_file, eval_batch_size, _
↪clsid2catid):
    batch_im_id_list, batch_im_name_list, batch_img_bytes_list = get_image_as_
↪bytes(images, eval_pre_path)

    # warm up
    yolo_predictor({'image': np.array(batch_img_bytes_list[0], dtype=object)})

    with futures.ThreadPoolExecutor(4) as exe:
        fut_im_list = []
        fut_list = []
        start_time = time.time()
        for batch_im_id, batch_im_name, batch_img_bytes in zip(batch_im_id_list, batch_
↪im_name_list, batch_img_bytes_list):
            if len(batch_img_bytes) != eval_batch_size:
                continue
            fut = exe.submit(yolo_predictor, {'image': np.array(batch_img_bytes,␣
↪dtype=object)})
            fut_im_list.append((batch_im_id, batch_im_name))
            fut_list.append(fut)
        bbox_list = []
        count = 0
        for (batch_im_id, batch_im_name), fut in zip(fut_im_list, fut_list):
            results = fut.result()
            bbox_list.extend(analyze_bbox(results, batch_im_id, _clsid2catid))
            for _ in batch_im_id:
                count += 1
                if count % 100 == 0:
                    print('Test iter {}'.format(count))
        print('==================== Performance Measurement ====================')
        print('Finished inference on {} images in {} seconds'.format(len(images), time.
↪time() - start_time))
        print('=================================================================')
    # start evaluation
```

(continues on next page)

(continued from previous page)

```
    box_ap_stats = bbox_eval(anno_file, bbox_list)
    return box_ap_stats
```

Evaluate mean average precision (mAP) score

Here is the code to calculate mAP scores of the YOLO v3 model. The expected mAP score is around 0.328 if we use the pretrained weights.

```
[ ]: yolo_pred = tf.contrib.predictor.from_saved_model('./yolo_v3_coco_saved_model_neuron')

val_coco_root = './val2017'
val_annotate = './annotations/instances_val2017.json'
clsid2catid = {0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 10, 10: 11, 11:␣
↪13, 12: 14, 13: 15, 14: 16,
               15: 17, 16: 18, 17: 19, 18: 20, 19: 21, 20: 22, 21: 23, 22: 24, 23: 25,␣
↪24: 27, 25: 28, 26: 31,
               27: 32, 28: 33, 29: 34, 30: 35, 31: 36, 32: 37, 33: 38, 34: 39, 35: 40,␣
↪36: 41, 37: 42, 38: 43,
               39: 44, 40: 46, 41: 47, 42: 48, 43: 49, 44: 50, 45: 51, 46: 52, 47: 53,␣
↪48: 54, 49: 55, 50: 56,
               51: 57, 52: 58, 53: 59, 54: 60, 55: 61, 56: 62, 57: 63, 58: 64, 59: 65,␣
↪60: 67, 61: 70, 62: 72,
               63: 73, 64: 74, 65: 75, 66: 76, 67: 77, 68: 78, 69: 79, 70: 80, 71: 81,␣
↪72: 82, 73: 84, 74: 85,
               75: 86, 76: 87, 77: 88, 78: 89, 79: 90}
eval_batch_size = 8
with open(val_annotate, 'r', encoding='utf-8') as f2:
    for line in f2:
        line = line.strip()
        dataset = json.loads(line)
        images = dataset['images']
box_ap = evaluate(yolo_pred, images, val_coco_root, val_annotate, eval_batch_size,␣
↪clsid2catid)
```

This document is relevant for: `Inf1`

9.8 Running SSD300 with AWS Neuron

*Update 11/16: The model checkpoint link*https://api.ngc.nvidia.com/v2/models/nvidia/ssdpyt_fp32/versions/1/files/nvidia_ssdpyt_fp32_20190225.pt*is currently broken and the AWS Neuron team is working on providing an alternative source.*

This demo shows a Neuron compatible SSD300 implementation that is functionally equivalent to open source SSD300 model. This demo uses TensorFlow-Neuron, PyTorch SSD300 model and checkpoint (https://pytorch.org/hub/nvidia_deeplearningexamples_ssd/) and also shows the performance achieved by the Inf1 instance.

9.8.1 Table of Contents

1. Launch EC2 instance and update AWS Neuron SDK software
2. Generating Neuron compatible SSD300 TensorFlow SavedModel
   - Convert open source PyTorch SSD300 model and checkpoint into Neuron compatible SSD300 TensorFlow SavedModel
3. Evaluate the generated SSD300 TensorFlow SavedModel for both accuracy and performance
   - Running threaded inference through the COCO 2017 validation dataset

9.8.2 Launch EC2 instances and update tensorflow-neuron and neuron-cc

For this demo, launch one inf1.xlarge EC2 instance. We recommend using the latest Ubuntu 18 Deep Learning AMI (DLAMI).

Please configure your ubuntu16/ubuntu18/yum repo following the steps in the *Install TensorFlow Neuron* in order to install `tensorflow-model-server-neuron`.

9.8.3 Generating Neuron compatible SSD300 TensorFlow SavedModel

First connect to your inf1.xlarge instance

Compile open source PyTorch SSD300 model and checkpoint into Neuron compatible SSD300 TensorFlow SavedModel

In the same directory ssd300_demo, run the following:

1. Create venv and install dependencies

```
sudo apt update
sudo apt install g++ python3-dev python3-venv unzip
sudo apt install tensorflow-model-server-neuron
python3 -m venv env
source ./env/bin/activate
pip install pip setuptools --upgrade
pip install -r ./requirements.txt --extra-index-url=https://pip.repos.neuron.amazonaws.
↪com
```

2. Clone NVIDIA's DeepLearningExamples repo that contains PyTorch SSD300.

```
git clone https://github.com/NVIDIA/DeepLearningExamples.git
cd DeepLearningExamples
git checkout a644350589f9abc91b203f73e686a50f5d6f3e96
cd ..
```

3. Download PyTorch SSD300 checkpoint file.

```
curl -LO https://api.ngc.nvidia.com/v2/models/nvidia/ssdpyt_fp32/versions/1/files/nvidia_
↪ssdpyt_fp32_20190225.pt
```

4. Download COCO 2017 validation set and annotations.

```
curl -LO http://images.cocodataset.org/zips/val2017.zip
unzip ./val2017.zip
curl -LO http://images.cocodataset.org/annotations/annotations_trainval2017.zip
unzip ./annotations_trainval2017.zip
```

5. Convert PyTorch SSD300 model and checkpoint into a Neuron-compatible TensorFlow SavedModel.

```
python ssd300_model.py --torch_checkpoint=./nvidia_ssdpyt_fp32_20190225.pt --output_
↪saved_model=./ssd300_tf_neuron/1
```

This converts PyTorch SSD300 model and checkpoint to a Neuron-compatible TensorFlow SavedModel using tensorflow-neuron and neuron-cc. The compilation output is stored in `./ssd300_tf_neuron`.

6. Launch the `tensorflow-model-server-neuron` gRPC server at default port 8500 in the background.

```
tensorflow_model_server_neuron --model_base_path=$(pwd)/ssd300_tf_neuron &
```

7. In client, evaluate the Neuron-compatible TensorFlow SavedModel for both accuracy and performance. Note that this client by default assumes a `tensorflow-model-server-neuron` listening at `localhost:8500`. On inf1.xlarge, the expected throughput is 100 images/second once the server is fully warmed up, and the expected mean average precision (mAP) is 0.253.

```
python ssd300_evaluation_client.py --val2017=./val2017 --instances_val2017_json=./
↪annotations/instances_val2017.json
```

8. After running the demo, please cleanup resources allocated in Neuron runtime by gracefully killing the `tensorflow_model_server_neuron` process, e. g.,

```
killall tensorflow_model_server_neuron
```

This document is relevant for: `Inf1`

9.9 Tensorflow ResNet 50 Optimization Tutorial

9.9.1 Note: this tutorial runs on tensorflow-neuron 1.x only

9.9.2 Introduction:

In this tutorial we provide three main sections:

- Take a Resnet 50 model and perform optimizations on it
- Compile the model with different batch sizes and Neuroncore Group sizes (read about Neuroncore Group sizes here: https://awsdocs-neuron.readthedocs-hosted.com/en/latest/neuron-guide/neuron-runtime/nrt-theory-of-operation.html#neuron-core-group)
- Run inference on our multiple compiled models to see which has the best throughput

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the Tensorflow Installation Guide. You can select the Kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

9.9.3 Install Dependencies

[ ]:
```
!pip install pillow requests # Necessary for loading images
!pip install 'tensorflow-neuron<2' --extra-index-url=https://pip.repos.neuron.amazonaws.
↪com
```

9.9.4 Compile

The following example shows how to compile a FP16 ResNet50 network using various batching parameters to find the optimal solution. On inf1.6xlarge, run through the following steps to get a optimized Resnet 50 model. First, extract Keras ResNet50 FP32 (resnet50_fp32_keras.pb will be generated):

[ ]:
```
import re
import argparse
import tensorflow as tf
import numpy as np

from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions

from google.protobuf import text_format
import tensorflow.python.saved_model

# set Keras global configurations
tf.keras.backend.set_learning_phase(0)
tf.keras.backend.set_image_data_format('channels_last')

float_type = 'float32'
float_type2 = 'fp32'
tf.keras.backend.set_floatx(float_type)

# load pre-trained model using Keras
model_name = 'resnet50_%s_keras'%float_type2
model = ResNet50(weights='imagenet')

# various save files
frozen_file = model_name + '.pb'
opt_file = model_name + '_opt.pb'

# obtain parameters
model_input = model.input.name.replace(':0', '')
model_output = model.output.name.replace(':0', '')
batch, height, width, channels = model.input.shape

print ("model, frozen file, optimized file, input size, input node, output node,")
print ("%s, %s, %s, %dx%dx%d, %s, %s" %(model_name, frozen_file, opt_file, width,␣
↪height, channels, model_input, model_output) )

# obtain the TF session
sess = tf.compat.v1.keras.backend.get_session()
```

(continues on next page)

(continued from previous page)

```
# save checkpoint files for freeze_graph
ckpt_file = '/tmp/' + model_name + '/' + model_name + '.ckpt'
graph_file = '/tmp/' + model_name + '/' + model_name + '.pb'
tf.compat.v1.train.Saver().save(sess, ckpt_file)
tf.io.write_graph(sess.graph.as_graph_def(), logdir='.', name=graph_file, as_text=False)

print(model_output)
with tf.compat.v1.Session(graph=tf.Graph()) as sess:
      saver = tf.compat.v1.train.import_meta_graph(ckpt_file + '.meta')
      saver.restore(sess, ckpt_file)
      output_graph_def = tf.compat.v1.graph_util.convert_variables_to_constants(
          sess, tf.compat.v1.get_default_graph().as_graph_def(), [model_output])
      output_graph_def = tf.compat.v1.graph_util.remove_training_nodes(
          output_graph_def, protected_nodes=[model_output])
      with open(frozen_file, 'wb') as f:
          f.write(output_graph_def.SerializeToString())
```

Optimize the extracted Keras ResNet50 FP32 graph for inference before casting (resnet50_fp32_keras_opt.pb will be generated) with the following transformations to the graph:

- Remove Identity and CheckNumerics nodes
- Fold FusedBatchNorm constants into previous Conv2D weights
- Fold other constants
- Strip unused nodes
- Sort by execution order

[ ]:
```
import copy
import string

from google.protobuf import text_format
from tensorflow.core.framework import node_def_pb2
from tensorflow.core.framework import attr_value_pb2
from tensorflow.python.framework import tensor_util
from tensorflow.tools.graph_transforms import TransformGraph

def clear_input(node):
  for i in range(len(node.input)):
    node.input.pop()

def replace_name(node, name):
  node.name = name

def replace_input(node, input_name, new_name):
  # node.input.replace(input_name, new_name)
  temp = []
  for i in node.input:
    temp.extend([new_name if i == input_name else i])
  clear_input(node)
  for i in temp:
    node.input.extend([i])
```

(continues on next page)

(continued from previous page)

```
def swap_names(node1, node2):
  temp = node2.name
  node2.name = node1.name
  node1.name = temp

def get_const_node(const_node_name, const_by_name):
  name = re.sub("/read$", "", const_node_name)
  return const_by_name[name]

def get_const_ndarray(const_node_name, const_by_name):
  name = re.sub("/read$", "", const_node_name)
  node = const_by_name[name]
  return tf.make_ndarray(node.attr.get("value").tensor)

def adjust_bias_values(bias_node, fbn_node, const_by_name):
  bias_val = get_const_ndarray(bias_node.input[1], const_by_name)
  gamma_val = get_const_ndarray(fbn_node.input[1], const_by_name)
  mean_val = get_const_ndarray(fbn_node.input[3], const_by_name)
  variance_val = get_const_ndarray(fbn_node.input[4], const_by_name)
  new_bias = bias_val * gamma_val / np.sqrt(variance_val)
  new_tensor = tensor_util.make_tensor_proto(new_bias, new_bias.dtype, new_bias.shape)
  bias_const_node = get_const_node(bias_node.input[1], const_by_name)
  bias_const_node.attr["value"].CopyFrom(attr_value_pb2.AttrValue(tensor=new_tensor))

def MoveBiasAddAfterFusedBatchNorm(graphdef):
  """fold_batch_norm function of TransformGraph is unable to fold Keras ResNet50
  because of BiasAdd between Conv2D and FusedBatchNorm (BiasAdd is not needed
  if FusedBatchNorm is used, but it exists in Keras ResNet50). Here, we
  move BiasAdd to after FusedBatchNorm, and adjust bias value by gamma/sqrt(variance).
  """
  sess = tf.compat.v1.Session(graph=tf.import_graph_def(graphdef))
  output_graph_def = tf.compat.v1.GraphDef()
  node_by_name = {}
  const_by_name = {}
  for node in graphdef.node:
    # Hack: use FusedBatchNormV2 so fold_batch_norm can recognize
    if node.op == "FusedBatchNormV3":
      node.op = "FusedBatchNorm"
      del(node.attr["U"])
      #import pdb; pdb.set_trace()
    copied_node = node_def_pb2.NodeDef()
    copied_node.CopyFrom(node)
    node_by_name[node.name] = copied_node
    skip_add_node = False
    # Switch Mul/BiasAdd in Keras RN50 so fold_batch_norm transform would work
    if node.op == "Const":
      const_by_name[node.name] = copied_node
    elif node.op.startswith("FusedBatchNorm"):
      inputs = node.input
      for i in inputs:
        input_node = node_by_name[i]
        if input_node.op == "BiasAdd":
```

(continues on next page)

(continued from previous page)

```
          output_graph_def.node.remove(input_node)
          input_node_input0 = input_node.input[0]
          # Adjust bias values (multiply by scale/sqrt(variance))
          adjust_bias_values(input_node, node, const_by_name)
          # Hack: swap names to avoid changing input of activation
          swap_names(copied_node, input_node)
          # Fix inputs for these two ops
          replace_input(copied_node, i, input_node_input0)
          replace_input(input_node, input_node_input0, copied_node.name)
          # Fix order in node list
          output_graph_def.node.extend([copied_node])
          output_graph_def.node.extend([input_node])
          skip_add_node = True
    # Add maybe-modified nodes if not already done
    if not skip_add_node:
      output_graph_def.node.extend([copied_node])
  return output_graph_def

def FoldFusedBatchNorm(graph_def):
  """Optimize training graph for inference:
    - Remove Identity and CheckNumerics nodes
    - Fold FusedBatchNorm constants into previous Conv2D weights
    - Fold other constants
    - Strip unused nodes
    - Sort by execution order
  """
  transformed_graph_def = TransformGraph (
         graph_def,
         ['input_1'],
         ['probs/Softmax'],
         [
            'add_default_attributes',
            'remove_nodes(op=Identity, op=CheckNumerics)',
            'fold_constants(ignore_errors=true)',
            'fold_batch_norms',
            'fold_old_batch_norms',
            'strip_unused_nodes',
            'sort_by_execution_order',
         ])
  return transformed_graph_def

def load_graph(model_file):
  graph_def = tf.compat.v1.GraphDef()

  with open(model_file, "rb") as f:
    graph_def.ParseFromString(f.read())
  return graph_def


graph_orig = load_graph('resnet50_fp32_keras.pb')
graph_mod = MoveBiasAddAfterFusedBatchNorm(graph_orig)
graph_mod2 = FoldFusedBatchNorm(graph_mod)
```

(continues on next page)

(continued from previous page)

```
with tf.io.gfile.GFile('resnet50_fp32_keras_opt.pb', "wb") as f:
    f.write(graph_mod2.SerializeToString())
```

Convert full graph to FP16 (resnet50_fp16_keras_opt.pb will be generated. This will take about a minute.

```
[ ]: from tensorflow.core.framework import graph_pb2
     from tensorflow.python.platform import gfile

     def ConvertFP32ToOther(graphdef):
       """Converts an FP32 network by casting all constants (weights) to a lower
          precision floating point type (FP16) and updating the dtypes
          everywhere."""
       cast_type = "float16"
       sess = tf.Session(graph=tf.import_graph_def(graphdef))
       output_graph_def = graph_pb2.GraphDef()
       dummy_tensor = sess.run(tf.constant([0.1]))
       dummy_tensor_proto = tensor_util.make_tensor_proto(dummy_tensor, \
           dtype=cast_type, shape=dummy_tensor.shape)
       dummy_tensor32 = sess.run(tf.constant([0.1]))
       dummy_tensor_proto32 = tensor_util.make_tensor_proto(dummy_tensor, \
           dtype=tf.float32, shape=dummy_tensor.shape)
       dt_float_type_attr = attr_value_pb2.AttrValue(type=dummy_tensor_proto32.dtype)
       dt_half_type_attr = attr_value_pb2.AttrValue(type=dummy_tensor_proto.dtype)
       for node in graphdef.node:
         output_node = node_def_pb2.NodeDef()
         output_node.CopyFrom(node)
         if (node.op == "Const"):
           if (node.attr["dtype"] == dt_float_type_attr):
             a = tensor_util.MakeNdarray(node.attr["value"].tensor)
             a = tf.cast(a, cast_type)
             a = sess.run(a)
             output_node.attr["dtype"].CopyFrom(dt_half_type_attr)
             output_node.attr["value"].CopyFrom(
                 attr_value_pb2.AttrValue(
                   tensor=tensor_util.make_tensor_proto(a,\
                     dtype=cast_type, shape=a.shape)))
         else:
           if ("T" in node.attr.keys()):
             if (output_node.attr["T"] == dt_float_type_attr):
               output_node.attr["T"].CopyFrom(dt_half_type_attr)
           if ("Tparams" in node.attr.keys()):
             if (output_node.attr["Tparams"] == dt_float_type_attr):
               output_node.attr["Tparams"].CopyFrom(dt_half_type_attr)
           if ("dtype" in node.attr.keys()):
             if (node.attr["dtype"] == dt_float_type_attr):
               output_node.attr["dtype"].CopyFrom(dt_half_type_attr)
           if ("SrcT" in node.attr.keys()):
             if (node.attr["SrcT"] == dt_float_type_attr):
               output_node.attr["SrcT"].CopyFrom(dt_half_type_attr)
           if ("DstT" in node.attr.keys()):
             if (node.attr["DstT"] == dt_float_type_attr):
               output_node.attr["DstT"].CopyFrom(dt_half_type_attr)
```

(continues on next page)

(continued from previous page)

```
    output_graph_def.node.extend([output_node])
  return output_graph_def

def load_graph(model_file):
  graph_def = tf.GraphDef()

  with open(model_file, "rb") as f:
    graph_def.ParseFromString(f.read())

  return graph_def

graph_f32 = load_graph('resnet50_fp32_keras_opt.pb')
graph_f16 = ConvertFP32ToOther(graph_f32)
output_xformed_graph_name = 'resnet50_fp16_keras_opt.pb'
with gfile.GFile(output_xformed_graph_name, "wb") as f:
    f.write(graph_f16.SerializeToString())
```

Run the compilation script to sweep through various batch sizes up to 5 and several NeuronCore Group sizes up to 16. The script calls the compilation script pb2sm_compile.py which tries to perform compilation. Some error messages are expected due to known issues (see Known Issues section in the tutorial). If you run all the configurations it will take about 45 minutes.

[ ]:
```
%%bash
#!/usr/bin/env bash

echo "" > full_sweep.log
echo "" > full_sweep_results.txt

results=()
for b in $(seq 1 5); do
    for i in 1 2 4 8 12 16; do
        python pb2sm_compile.py --batch_size=$b --neuroncore-pipeline-cores=$i | tee -a␣
↪full_sweep.log;
        results[$b]+=", "`tail -1 full_sweep.log`
    done
done

head="batch"
for i in 1 2 4 8 12 16; do
    head+=", nc${i}"
done
echo $head | tee -a full_sweep_results.txt
for b in $(seq 1 5); do
    echo $b${results[$b]} | tee -a full_sweep_results.txt
done
```

You should see some output like this:

```
INFO: Compilation finished in 95 seconds with 99.5% operations placed on Inferentia

1
```

(continues on next page)

(continued from previous page)

```
*** Batch size 1, num NeuronCores 2 (input shape: (1, 224, 224, 3), saved model dir:␣
↪rn50_fp16_compiled_b1_nc2) ***

INFO: Compilation finished in 95 seconds with 99.5% operations placed on Inferentia

1

*** Batch size 1, num NeuronCores 4 (input shape: (1, 224, 224, 3), saved model dir:␣
↪rn50_fp16_compiled_b1_nc4) ***

INFO: Compilation finished in 95 seconds with 99.5% operations placed on Inferentia

1

... (outputs removed)

*** Batch size 5, num NeuronCores 16 (input shape: (5, 224, 224, 3), saved model dir:␣
↪rn50_fp16_compiled_b5_nc16) ***

ERROR: Compilation finished in 120 seconds with less than 50% operations placed on␣
↪Inferentia (0.0%)

INFO: Retry compilation without static weights

ERROR: Retry compilation finished in 137 seconds with less than 50% operations placed on␣
↪Inferentia (0.0%)

0

The file full_sweep_results.txt shows a summary of the sweep results with latest Neuron␣
↪1/27/20 release (0 means compilation unsuccessful and 0 ops mapped to Inferentia, 1␣
↪means most ops mapped to Inferentia and non-static weights, 2 means most ops mapped to␣
↪Inferentia and using static weights):

batch, nc1, nc2, nc4, nc8, nc12, nc16
1, 1, 1, 1, 2, 2, 2
2, 1, 1, 0, 1, 2, 2
3, 1, 1, 1, 1, 1, 1
4, 1, 1, 0, 1, 1, 1
5, 1, 1, 0, 0, 0, 0
```

9.9.5 Inference

Run inference over different batch sizes and Neuroncore groups to obtain throughput and latency results for ResNet50. To apply dynamic batching, the user batch size is set to 10x the compiled batch size, in order to keep input queue full and to amortize framework-to-Neuron overhead.

Note: The results are based on the Neuron v1.12.2 (Mar 4th 2021) release. These will continue improve as we increase Neuron performance.

[ ]:
```
!cd ~/aws-neuron-sdk/src/examples/tensorflow/keras_resnet50/
!echo "" > batch.log
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=1 | tee -a batch.log; done
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=2 | tee -a batch.log; done
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=4 | tee -a batch.log; done
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=8 | tee -a batch.log; done
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=12 | tee -a batch.log; done
!for i in $(seq 1 5); do python infer_resnet50_keras_loadtest.py --batch_size=$i --
↪neuroncore-pipeline-cores=16 | tee -a batch.log; done
```

The file batch.log now contains the results for each batch size. We can look at the throughput values to get an idea of which models are performing well. The output should look something like this:

The model best model configuration for throughput (if you run on an Inf1.6xlarge as suggested in the tutorial) is batch size 5 NeuronCore group size 2. Increasing batch size usually helps to increase throughput (up to a certain extent).

```
*** Compiled batch size 5, user batch size 10, num NeuronCores 2 (input shape: (10, 224,␣
↪224, 3), saved model dir: ./rn50_fp16_compiled_b5_nc2/1) ***

Instance type inf1.6xlarge with 16 NeuronCores
NEURON_MAX_NUM_INFERS (env): 5
NEURONCORE_GROUP_SIZES (env): 2,2,2,2,2,2,2,2
NUM THREADS:  16
NUM_LOOPS_PER_THREAD:  400
USER_BATCH_SIZE:  10
Throughput values collected:
[10680, 10700, 10660]

(rest of outputs removed)
```

9.9.6 Known Issues

Unable to compile with batch and num NeuronCores combination

For some combination of batch and number of NeuronCores setting, you may see an internal compiler error as below. Please see the sweep result above for Neuron 1/27/20 release. Furthermore, if using auto-casting to bfloat16 from FP32 network and batch size is larger than 1 would result in the same error.

```
INFO:tensorflow:fusing subgraph neuron_op_a73aed4b95ca5d5b with neuron-cc; log file is␣
↪at /home/ubuntu/keras_fp16_benchmarking_db/compiler_workdir/neuron_op_a73aed4b95ca5d5b/
↪graph_def.neuron-cc.log
  WARNING:tensorflow:Failed to fuse subgraph neuron_op_a73aed4b95ca5d5b with '/home/
↪ubuntu/test_venv/bin/neuron-cc compile /home/ubuntu/keras_fp16_benchmarking_db/
↪compiler_workdir/neuron_op_a73aed4b95ca5d5b/graph_def.pb --framework TENSORFLOW --
↪pipeline compile SaveTemps --output /home/ubuntu/keras_fp16_benchmarking_db/compiler_
↪workdir/neuron_op_a73aed4b95ca5d5b/graph_def.neff --io-config "{\"inputs\": {\"input_
↪10/_0:0\": [[6, 224, 224, 3], \"float16\"]}, \"outputs\": [\"probs/Softmax:0\"]}" --
↪batching_en --rematerialization_en --sb_size 120 --spill_dis --enable-replication True'
  WARNING:tensorflow:neuron-cc error message:
  WARNING:tensorflow:01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:␣
↪*****************************************************************
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:  An Internal Compiler Error has occurred
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:␣
↪*****************************************************************
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Please contact Customer Support and provide␣
↪the following details.
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Error message:  Non-zero exit status (134)␣
↪for command: /home/ubuntu/test_venv/lib/python3.6/site-packages/neuroncc/starfish/bin/
↪list_sch --hhir hh-tr-external-move.json --verbose 0 --sb_size 120 --arith_intensity_
↪target 2300 --sb_watermark_low 0.250000 --sb_watermark_high 0.750000 --sb_size_tol 1 --
↪alloc simple1 --alloc_opt --depth_diff 0.100000 --verbose_start_cycle 0 --tt_dist --mm_
↪meet_cnt 1 --load_speed_factor 0.300000 --schir sch_tmp.json --spill_depth_limit 5 --
↪spill_dis --true_dep --mm_order --batching_en --rematerialization_en
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Error class:    CompilerInternalError
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Error location: job.Scheduler.3
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Command line:   /home/ubuntu/test_venv/bin/
↪neuron-cc compile /home/ubuntu/keras_fp16_benchmarking_db/compiler_workdir/neuron_op_
↪a73aed4b95ca5d5b/graph_def.pb --framework TENSORFLOW --pipeline compile SaveTemps --
↪output /home/ubuntu/keras_fp16_benchmarking_db/compiler_workdir/neuron_op_
↪a73aed4b95ca5d5b/graph_def.neff --io-config '{"inputs": {"input_10/_0:0": [[6, 224,␣
↪224, 3], "float16"]}, "outputs": ["probs/Softmax:0"]}' --batching_en --
↪rematerialization_en --sb_size 120 --spill_dis --enable-replication True
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Internal details:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:   File "neuroncc/driver/Job.py", line 207,␣
↪in neuroncc.driver.Job.runSingleInputFn
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:   File "neuroncc/driver/jobs/Scheduler.py",␣
↪line 58, in neuroncc.driver.jobs.Scheduler.Scheduler.runSingleInput
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:   File "neuroncc/driver/Job.py", line 145,␣
↪in neuroncc.driver.Job.Job.shellCommand
  01/23/2020 01:15:40 AM ERROR [neuron-cc]:
  01/23/2020 01:15:40 AM ERROR [neuron-cc]: Version information:
  01/23/2020 01:15:41 AM ERROR [neuron-cc]:   Neuron Compiler version 1.0.6632.
↪0+6001610955
  01/23/2020 01:15:41 AM ERROR [neuron-cc]:
  01/23/2020 01:15:41 AM ERROR [neuron-cc]:   HWM version 1.0.839.0-6001300654
  01/23/2020 01:15:41 AM ERROR [neuron-cc]:   NEFF version 0.6
```

(continues on next page)

(continued from previous page)

```
01/23/2020 01:15:41 AM ERROR [neuron-cc]:   TVM version 1.0.1589.0+6001610955
01/23/2020 01:15:41 AM ERROR [neuron-cc]:   NumPy version 1.16.5
01/23/2020 01:15:41 AM ERROR [neuron-cc]:   MXNet not available
01/23/2020 01:15:41 AM ERROR [neuron-cc]:   TF version 1.15.0
01/23/2020 01:15:41 AM ERROR [neuron-cc]:
```

[ ]:

```
```

Natural Language Processing (NLP) Tutorials

- Tensorflow 1.x - Running TensorFlow BERT-Large with AWS Neuron *[html]*
- Tensorflow 2.x - HuggingFace Pipelines distilBERT with Tensorflow2 Neuron *[html]* [notebook]

This document is relevant for: `Inf1`

9.10 Running TensorFlow BERT-Large with AWS Neuron

This example shows a Neuron compatible BERT-Large implementation that is functionally equivalent to open source BERT-Large model. This demo uses TensorFlow-Neuron, BERT-Large weights fine tuned for MRPC and also shows the performance achieved by the Inf1 instance. For users who want to use public BERT SavedModels please also follow the steps described *Using public BERT SavedModels*.

9.10.1 Launch EC2 instances

For this demo, launch two EC2 instances :

- a c5.4xlarge instance for compiling the BERT-Large Model and
- an inf1.xlarge instance for running inference

For both of these instances choose the latest Ubuntu 18 Deep Learning AMI (DLAMI).

9.10.2 Compiling Neuron compatible BERT-Large

First connect to a c5.4xlarge instance and update tensorflow-neuron and neuron-cc

Update compilation EC2 instance

Update to the latest neuron software by executing the instructions at *Install TensorFlow Neuron*.

Note: if your tensorflow-neuron version on the inference instance is lower than 1.15.0.1.0.1333.0, you will need to run this demo on inf1.2xlarge instead of inf1.xlarge.

Compile open source BERT-Large saved model using Neuron compatible BERT-Large implementation

Neuron software works with TensorFlow saved models. Users should bring their own BERT-Large saved model for this section. This demo will run inference for the MRPC task and the saved model should be fine tuned for MRPC. Users who need additional help to fine-tune the model for MRPC or to create a saved model can refer to *Appendix 1*.

In the same environment and directory bert_demo scripts, run the following :

```
git clone https://github.com/aws/aws-neuron-sdk
cd ~/aws-neuron-sdk/src/examples/tensorflow/bert_demo/
export BERT_LARGE_SAVED_MODEL="/path/to/user/bert-large/savedmodel"
python bert_model.py --input_saved_model $BERT_LARGE_SAVED_MODEL --output_saved_model ./
↪bert-saved-model-neuron --batch_size=6 --aggressive_optimizations
```

This compiles BERT-Large pointed to by $BERT_LARGE_SAVED_MODEL for an input size of 128 and batch size of 6. The compilation output is stored in bert-saved-model-neuron. Copy this to your Inf1 instance for inferencing.

The bert_model.py script encapsulates all the steps necessary for this process. For details on what is done by bert_model.py please refer to *Appendix 2*.

9.10.3 Running the inference demo

Connect to your inf1.xlarge instance and update tensorflow-neuron, aws-neuron-runtime and aws-neuron-tools.

Update inference EC2 instance

Update to the latest neuron software by executing the instructions at *Install TensorFlow Neuron*.

Launching the BERT-Large demo server

Copy the compiled model (bert-saved-model-neuron) from your c5.4xlarge to your inf1.xlarge instance. Place the model in the same directory as the bert_demo scripts. Then from the same conda environment launch the BERT-Large demo server :

```
cd ~/aws-neuron-sdk/src/examples/tensorflow/bert_demo/
python bert_server.py --dir bert-saved-model-neuron --batch 6 --parallel 4
```

This loads 4 BERT-Large models, one into each of the 4 NeuronCores found in an inf1.xlarge instance. For each of the 4 models, the BERT-Large demo server opportunistically stitches together asynchronous requests into batch 6 requests. When there are insufficient pending requests, the server creates dummy requests for batching.

Wait for the bert_server to finish loading the BERT-Large models to Inferentia memory. When it is ready to accept requests it will print the inferences per second once every second. This reflects the number of real inferences only. Dummy requests created for batching are not credited to inferentia performance. Once the inferences are done you can send a keyboard interrupt to print out the average throughput of your run.

Sending requests to server from multiple clients

Wait until the bert demo server is ready to accept requests. Then on the same inf1.xlarge instance, launch a separate linux terminal. From the bert_demo directory execute the following commands :

```
source activate aws_neuron_tensorflow_p36
cd ~/aws-neuron-sdk/src/examples/tensorflow/bert_demo/
for i in {1..96}; do python bert_client.py --cycle 128 & done
```

This spins up 96 clients, each of which sends 128 inference requests.

Printing latency metrics

After all your requests have been sent to your server you can run the following command:

```
python latency_printer.py
```

9.10.4 Using public BERT SavedModels

We are now providing a compilation script that has better compatibility with various flavors of BERT SavedModels generated from https://github.com/google-research/bert. Here are the current limitations:

1. You did not change modeling.py
2. BERT SavedModel is generated using `estimator.export_saved_model`
3. BERT SavedModel uses fixed sequence length 128 (you may check by `saved_model_cli show --dir /path/to/user/bert/savedmodel --all`)
4. `neuron-cc` version is at least 1.0.12000.0
5. `aws-neuron-runtime` version is at least 1.0.7000.0
6. The `--batch_size` argument specified in this script is at most 4

Example usage is shown below:

```
export BERT_LARGE_SAVED_MODEL="/path/to/user/bert-large/savedmodel"
cd ~/aws-neuron-sdk/src/examples/tensorflow/bert_demo/
python bert_no_model.py --input_saved_model $BERT_LARGE_SAVED_MODEL --output_saved_model␣
↪./bert-saved-model-neuron --batch_size=1
```

9.10.5 Appendix 1

Users who need help finetuning BERT-Large for MRPC and creating a saved model may follow the instructions here.

Connect to the c5.4xlarge compilation EC2 instance you started above and download these three items :

1. clone this github repo.
2. download GLUE data as described here. Do not run the finetuning command.
3. download a desired pre-trained BERT-Large checkpoint from here. This is the model we will fine tune.

Next edit run_classifier.py in the cloned bert repo to apply the patch described in the following git diff.

```
diff --git a/run_classifier.py b/run_classifier.py
index 817b147..c9426bc 100644
--- a/run_classifier.py
+++ b/run_classifier.py
@@ -955,6 +955,18 @@ def main(_):
         drop_remainder=predict_drop_remainder)

     result = estimator.predict(input_fn=predict_input_fn)
+    features = {
+        "input_ids": tf.placeholder(shape=[None, FLAGS.max_seq_length], dtype=tf.int32,␣
↪name='input_ids'),
+        "input_mask": tf.placeholder(shape=[None, FLAGS.max_seq_length], dtype=tf.int32,
↪ name='input_mask'),
+        "segment_ids": tf.placeholder(shape=[None, FLAGS.max_seq_length], dtype=tf.
↪int32, name='segment_ids'),
+        "label_ids": tf.placeholder(shape=[None], dtype=tf.int32, name='label_ids'),
+        "is_real_example": tf.placeholder(shape=[None], dtype=tf.int32, name='is_real_
↪example'),
+    }
+    serving_input_fn = tf.estimator.export.build_raw_serving_input_receiver_fn(features)
+    estimator._export_to_tpu = False  ## !!important to add this
+    estimator.export_saved_model(
+        export_dir_base='./bert_classifier_saved_model',
+        serving_input_receiver_fn=serving_input_fn)

     output_predict_file = os.path.join(FLAGS.output_dir, "test_results.tsv")
     with tf.gfile.GFile(output_predict_file, "w") as writer:
```

NOTE : Users who are interested may refer to this link for additional background information on the patch but it is not necessary for running this demo.

Then from the bert_demo directory run the following :

```
source activate aws_neuron_tensorflow_p36
cd ~/aws-neuron-sdk/src/examples/tensorflow/bert_demo/
export BERT_REPO_DIR="/path/to/cloned/bert/repo/directory"
export GLUE_DIR="/path/to/glue/data/directory"
export BERT_BASE_DIR="/path/to/pre-trained/bert-large/checkpoint/directory"
./tune_save.sh
```

The a saved model will be created in $BERT_REPO_DIR/bert-saved-model/*random_number*/. Where, *random_number* is a random number generated for every run. Use this saved model to continue with the rest of the demo.

9.10.6 Appendix 2

For all BERT variants, we currently need to augment the standard Neuron compilation process for performance tuning. In the future, we intend to automate this tuning process. This would allow users to use the standard Neuron compilation process, which requires only a one line change in user source code. The standard compilation process is described *Running Neuron Apache MXNet (Incubating) ResNet50 on Inferentia*.

The augmented Neuron compilation process is encapsulated by the bert_model.py script, which performs the following things :

1. Define a Neuron compatible implementation of BERT-Large. For inference, this is functionally equivalent to the open source BERT-Large. The changes needed to create a Neuron compatible BERT-Large implementation is described in *Appendix 3*.
2. Extract BERT-Large weights from the open source saved model pointed to by –input_saved_model and associates it with the Neuron compatible model
3. Invoke TensorFlow-Neuron to compile the Neuron compatible model for Inferentia using the newly associated weights
4. Finally, the compiled model is saved into the location given by –output_saved_model

9.10.7 Appendix 3

The Neuron compatible implementation of BERT-Large is functionally equivalent to the open source version when used for inference. However, the detailed implementation does differ and here are the list of changes :

1. Data Type Casting : If the original BERT-Large an FP32 model, bert_model.py contains manually defined cast operators to enable mixed-precision. FP16 is used for multi-head attention and fully-connected layers, and fp32 everywhere else. This will be automated in a future release.
2. Remove Unused Operators: A model typically contains training operators that are not used in inference, including a subset of the reshape operators. Those operators do not affect inference functionality and have been removed.
3. Reimplementation of Selected Operators : A number of operators (mainly mask operators), has been reimplemented to bypass a known compiler issue. This will be fixed in a planned future release.
4. Manually Partition Embedding Ops to CPU : The embedding portion of BERT-Large has been partitioned manually to a subgraph that is executed on the host CPU, without noticable performance impact. In near future, we plan to implement this through compiler auto-partitioning without the need for user intervention.

This document is relevant for: `Inf1`

9.11 Compiling and Deploying Pretrained HuggingFace Pipelines distilBERT with Tensorflow2 Neuron

9.11.1 Introduction

In this tutorial you will compile and deploy distilBERT version of HuggingFace Transformers BERT for Inferentia. The full list of HuggingFace's pretrained BERT models can be found in the BERT section on this page https://huggingface.co/transformers/pretrained_models.html. you can also read about HuggingFace's pipeline feature here: https://huggingface.co/transformers/main_classes/pipelines.html

This Jupyter notebook should be run on an instance which is inf1.6xlarge or larger, but in real life scenario the compilation should be done on a compute instance and the deployment on inf1 instance to save costs.

9.11.2 Setting up your environment:

To run this tutorial, please make sure you deactivate any existing TensorFlow conda environments you already using. Install TensorFlow 2.x by following the instructions at TensorFlow Tutorial Setup Guide.

After following the Setup Guide, you need to change your kernel to `Python (Neuron TensorFlow 2)` by clicking Kerenel->Change Kernel->`Python (Neuron TensorFlow 2)`

Now you can install TensorFlow Neuron 2.x, HuggingFace transformers, and HuggingFace datasets dependencies here.

[ ]:
```
!pip install --upgrade "transformers==4.1.0"
!pip install ipywidgets
```

[ ]:
```
from transformers import pipeline
import tensorflow as tf
import tensorflow.neuron as tfn
```

9.11.3 Compile the model into an AWS Neuron Optimized Model

[ ]:
```
#Create the huggingface pipeline for sentiment analysis
#this model tries to determine of the input text has a positive
#or a negative sentiment.
model_name = 'distilbert-base-uncased-finetuned-sst-2-english'

pipe = pipeline('sentiment-analysis', model=model_name, framework='tf')

#pipelines are extremely easy to use as they do all the tokenization,
#inference and output interpretation for you.
pipe(['I love pipelines, they are very easy to use!', 'this string makes it batch size␣
↪two'])
```

As yo've seen above, Huggingface's pipline feature is a great wrapper for running inference on their models. It takes care of the tokenization of the string inputs. Then feeds that tokenized input to the model. Finally it interprets the outputs of the model and formats them in a way that is very human readable. Our goal will be to compile the underlying model inside the pipeline as well as make some edits to the tokenizer. The reason you need to edit the tokenizer is to make sure that you have a standard sequence length (in this case 128) as neuron only accepts static input shapes.

[ ]:
```
neuron_pipe = pipeline('sentiment-analysis', model=model_name, framework='tf')

#the first step is to modify the underlying tokenizer to create a static
#input shape as inferentia does not work with dynamic input shapes
original_tokenizer = pipe.tokenizer


#you intercept the function call to the original tokenizer
#and inject our own code to modify the arguments
def wrapper_function(*args, **kwargs):
    kwargs['padding'] = 'max_length'
    #this is the key line here to set a static input shape
    #so that all inputs are set to a len of 128
    kwargs['max_length'] = 128
    kwargs['truncation'] = True
```

(continues on next page)

(continued from previous page)

```
    kwargs['return_tensors'] = 'tf'
    return original_tokenizer(*args, **kwargs)

#insert our wrapper function as the new tokenizer as well
#as reinserting back some attribute information that was lost
#when you replaced the original tokenizer with our wrapper function
neuron_pipe.tokenizer = wrapper_function
neuron_pipe.tokenizer.decode = original_tokenizer.decode
neuron_pipe.tokenizer.mask_token_id = original_tokenizer.mask_token_id
neuron_pipe.tokenizer.pad_token_id = original_tokenizer.pad_token_id
neuron_pipe.tokenizer.convert_ids_to_tokens = original_tokenizer.convert_ids_to_tokens

#Our example data!
string_inputs = [
    'I love to eat pizza!',
    'I am sorry. I really want to like it, but I just can not stand sushi.',
    'I really do not want to type out 128 strings to create batch 128 data.',
    'Ah! Multiplying this list by 32 would be a great solution!',
]
string_inputs = string_inputs * 32


example_inputs = neuron_pipe.tokenizer(string_inputs)
#compile the model by calling tfn.trace by passing in the underlying model
#and the example inputs generated by our updated tokenizer
neuron_model = tfn.trace(pipe.model, example_inputs)

#now you can insert the neuron_model and replace the cpu model
#so now you have a huggingface pipeline that uses and underlying neuron model!
neuron_pipe.model = neuron_model
neuron_pipe.model.config = pipe.model.config
```

9.11.4 Why use batch size 128?

You'll notice that in the above example we passed a two tensors of shape 128 (the batch size) x 128 (the sequence length) in this function call `tfn.trace(pipe.model, example_inputs)`. The example_inputs argument is important to `tfn.trace` because it tells the neuron model what to expect (remember that a neuron model needs static input shapes, so example_inputs defines that static input shape). A smaller batch size would also compile, but a large batch size ensures that the neuron hardware will be fed enough data to be as performant as possible.

9.11.5 What if my model isn't a Huggingface pipeline?

Not to worry! There is no requirement that your model needs to be Huggingface pipeline compatible. The Huggingface pipeline is just a wrapper for an underlying TensorFlow model (in our case `pipe.model`). As long as you have a TensorFlow 2.x model you can compile it on neuron by calling `tfn.trace(your_model, example_inputs)`. The processing the input and output to your own model is up to you! Take a look at the example below to see what happens when we call the model without the Huggingface pipeline wrapper as opposed to with it.

[ ]:
```
#directly call the model
print(neuron_model(example_inputs))
```

(continues on next page)

(continued from previous page)

```
#with the model inserted to the wrapper
print(neuron_pipe(string_inputs))

#Look at the difference between string_inputs
#and example_inputs

print(example_inputs)
print(string_inputs)
```

9.11.6 Save your neuron model to disk and avoid recompilation.

To avoid recompiling the model before every deployment, you can save the neuron model by calling `model_neuron.save(model_dir)`. This `save` method prefers to work on a flat input/output lists and does not work on dictionary input/output - which is what the Huggingface distilBERT expects as input. You can work around this by writing a simple wrapper that takes in an input list instead of a dictionary, compile the wrapped model and save it for later use.

[ ]:
```
class TFBertForSequenceClassificationFlatIO(tf.keras.Model):
    def __init__(self, model):
        super().__init__()
        self.model = model
    def call(self, inputs):
        input_ids, attention_mask = inputs
        output = self.model({'input_ids': input_ids, 'attention_mask': attention_mask})
        return output['logits']

#wrap the original model from HuggingFace, now our model accepts a list as input
model_wrapped = TFBertForSequenceClassificationFlatIO(pipe.model)
#turn the dictionary input into list input
example_inputs_list = [example_inputs['input_ids'], example_inputs['attention_mask']]

#compile the wrapped model and save it to disk
model_wrapped_traced = tfn.trace(model_wrapped, example_inputs_list)
model_wrapped_traced.save('./distilbert_b128')
```

[ ]:
```
!ls #you should now be able to see the model saved as the folder distilbert_b128
```

9.11.7 Load the model from disk

Now you can reload the model by calling `tf.keras.models.load_model(str : model_directory)`. This model is already compiled and could run inference on neuron, but if you want it to work with our Huggingface pipeline, you have to wrap it again to accept dictionary input.

[ ]:
```
class TFBertForSequenceClassificationDictIO(tf.keras.Model):
    def __init__(self, model_wrapped):
        super().__init__()
        self.model_wrapped = model_wrapped
        self.aws_neuron_function = model_wrapped.aws_neuron_function
    def call(self, inputs):
        input_ids = inputs['input_ids']
```

(continues on next page)

(continued from previous page)

```
        attention_mask = inputs['attention_mask']
        logits = self.model_wrapped([input_ids, attention_mask])
        return [logits]


reloaded_model = tf.keras.models.load_model('./distilbert_b128')
rewrapped_model = TFBertForSequenceClassificationDictIO(model_wrapped_traced)

#now you can reinsert our reloaded model back into our pipeline
neuron_pipe.model = rewrapped_model
neuron_pipe.model.config = pipe.model.config
```

9.11.8 Benchmarking the neuron model

Now you can do some simple benchmarking of the neuron model. If you are running this tutorial on a inf1.6xlarge, as suggested, you must tell neuron to use all 16 Neuron Cores to get maximum throughput. By default, TensorFlow Neuron will use only one Inferentia chip, which has 4 Neuron Cores. An inf1.6xlarge has 4 Inferentia chips. To tell Neuron to run on all available cores, you can set the environment variable `NEURONCORE_GROUP_SIZES` or launch multiple processes that query the same model. To read more about this refer to the Neuron Core groups section of our documentation. Use a warmup inference on the neuron model before benchmarking, as the first inference call also loads the model onto inferentia.

[ ]:
```
import warnings

warnings.warn("NEURONCORE_GROUP_SIZES is being deprecated, if your application is using␣
↪NEURONCORE_GROUP_SIZES please \
see https://awsdocs-neuron.readthedocs-hosted.com/en/latest/release-notes/deprecation.
↪html#announcing-end-of-support-for-neuroncore-group-sizes \
for more details.", DeprecationWarning)
%env NEURONCORE_GROUP_SIZES='16x1'

import time

#warmup inf
neuron_pipe(string_inputs)
#benchmark batch 128 neuron model
neuron_b128_times = []
for i in range(1000):
    start = time.time()
    outputs = neuron_pipe(string_inputs)
    end = time.time()
    neuron_b128_times.append(end - start)


neuron_b128_times = sorted(neuron_b128_times)

print(f"Average throughput for batch 128 neuron model is {128/(sum(neuron_b128_times)/
↪len(neuron_b128_times))} sentences/s.")
print(f"Peak throughput for batch 128 neuron model is {128/min(neuron_b128_times)}␣
↪sentences/s.")
print()
```

(continues on next page)

(continued from previous page)

```
print(f"50th percentile latency for batch 128 neuron model is {neuron_b128_
↪times[int(1000*.5)] * 1000} ms.")
print(f"90th percentile latency for batch 128 neuron model is {neuron_b128_
↪times[int(1000*.9)] * 1000} ms.")
print(f"95th percentile latency for bacth 128 neuron model is {neuron_b128_
↪times[int(1000*.95)] * 1000} ms.")
print(f"99th percentile latency for batch 128 neuron model is {neuron_b128_
↪times[int(1000*.99)] * 1000} ms.")
print()
```

[ ]:

```
```

Utilizing Neuron Capabilities Tutorials

- Tensorflow 1.x - Using NEURON_RT_VISIBLE_CORES with TensorFlow Serving *[html]*

This document is relevant for: `Inf1`

9.12 Using NEURON_RT_VISIBLE_CORES with TensorFlow Serving

TensorFlow serving allows customers to scale-up inference workloads across a network. TensorFlow Neuron Serving uses the same API as normal TensorFlow Serving with two differences: (a) the saved model must be compiled for Inferentia and (b) the entry point is a different binary named `tensorflow_model_server_neuron`. The binary is found at `/usr/local/bin/tensorflow_model_server_neuron` and is pre-installed in the DLAMI or installed with APT/YUM tensorflow-model-server-neuron package.

9.12.1 Install TensorFlow Model Server and Serving API

Follow the steps in the *Install TensorFlow Neuron*.

Then ensure you install using either apt-get or yum. If using TF 1.x, install the appropriate version (see above).:

```
sudo apt-get install tensorflow-model-server-neuron
```

or

```
sudo yum install tensorflow-model-server-neuron
```

Also, you would need TensorFlow Serving API (use –no-deps to prevent installation of regular tensorflow). Depending on the version of Tensorflow you wish to use:

For Tensorflow 1.x:

```
pip install --no-deps tensorflow_serving_api==1.15
```

For Tensorflow 2.x:

```
pip install --no-deps tensorflow_serving_api
```

For the example image preprocessing using Keras preprocessing, the Python Imaging Library Pillow is required:

```
pip install pillow
```

To workaround h5py issue https://github.com/aws/aws-neuron-sdk/issues/220:

```
pip install "h5py<3.0.0"
```

9.12.2 Export and Compile Saved Model

The following example shows graph construction followed by the addition of Neuron compilation step before exporting to saved model.

For Tensorflow 1.x:

```
import tensorflow as tf
import tensorflow.neuron

tf.keras.backend.set_learning_phase(0)
tf.keras.backend.set_image_data_format('channels_last')
model = tf.keras.applications.ResNet50(weights='imagenet')
sess = tf.keras.backend.get_session()
inputs = {'input': model.inputs[0]}
outputs = {'output': model.outputs[0]}

# save the model using tf.saved_model.simple_save
modeldir = "./resnet50/1"
tf.saved_model.simple_save(sess, modeldir, inputs, outputs)

# compile the model for Inferentia
neuron_modeldir = "./resnet50_inf1/1"
tf.neuron.saved_model.compile(modeldir, neuron_modeldir, batch_size=1)
```

For Tensorflow 2.x:

```
import tensorflow as tf
import tensorflow.neuron as tfn
import numpy as np

tf.keras.backend.set_learning_phase(0)
tf.keras.backend.set_image_data_format('channels_last')
image_sizes = [224, 224]
model = tf.keras.applications.ResNet50(weights='imagenet')
example_inputs = tf.random.uniform([1, *image_sizes, 3], dtype=tf.float32)

model_neuron = tfn.trace(model, example_inputs)
tf.keras.models.save_model(model_neuron, './resnet50_inf1/1')
```

9.12.3 Serving Saved Model

User can now serve the saved model with the tensorflow_model_server_neuron binary. To utilize multiple NeuronCores, it is recommended to launch multiple tensorflow model servers that listen to the same gRPC port:

```
export NEURON_RT_VISIBLE_CORES=0  # important to set this environment variable before␣
↪launching model servers
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
     --model_base_path=$(pwd)/resnet50_inf1/ --port=8500

#then to run another server on a different neuron core open another
#window and run this, except this time set NEURON_RT_VISIBLE_CORES=1
#you can keep doing this up to the number of Neuron Cores on your machine

export NEURON_RT_VISIBLE_CORES=1
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
     --model_base_path=$(pwd)/resnet50_inf1/ --port=8500
```

The compiled model is staged in Inferentia DRAM by the server to prepare for inference.

9.12.4 Generate inference requests to the model server

Now run inferences via GRPC as shown in the following sample client code:

For Tensorflow 1.x:

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.resnet50 import decode_predictions
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/
↪kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.contrib.util.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))
```

For Tensorflow 2.x:

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

tf.keras.backend.set_image_data_format('channels_last')

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/
↪kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input_1'].CopyFrom(
        tf.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output_1'])
    print(decode_predictions(prediction))
```

This document is relevant for: `Inf1`

Note: To use Jupyter Notebook see:

- setup-jupyter-notebook-steps-troubleshooting
- running-jupyter-notebook-as-script

Additional Examples (`tensorflow-neuron`)

- AWS Neuron Samples GitHub Repository

API Reference Guide (`tensorflow-neuron`)

This document is relevant for: `Inf1`

9.13 TensorFlow 2.x (tensorflow-neuron) Tracing API

The Neuron tracing API enables tracing TensorFlow 2.x models for deployment on AWS Machine Learning Accelerators.

9.13.1 Method

`tensorflow.neuron.trace`

9.13.2 Description

Trace a `keras.Model` or a Python callable that can be decorated by `tf.function`, and return an AWS-Neuron-optimized `keras.Model` that can execute on AWS Machine Learning Accelerators. Tracing is ideal for `keras.Model` that accepts a list of `tf.Tensor` objects and returns a list of `tf.Tensor` objects. It is expected that users will provide example inputs, and the `trace` function will execute `func` symbolically and convert it to a `keras.Model`.

The returned `keras.Model` will support inference only. Attributes or variables held by the original function or `keras.Model` will be dropped.

The returned `keras.Model` can be exported as SavedModel and served using TensorFlow Serving. Please see tensorflow-serving for more information about exporting to saved model and serving using TensorFlow Serving.

Options can be passed to Neuron compiler via the environment variable `NEURON_CC_FLAGS`. For example, the syntax `env NEURON_CC_FLAGS="--neuroncore-pipeline-cores=4"` directs Neuron compiler to compile each subgraph to fit in the specified number of NeuronCores. This number can be less than the total available NeuronCores on an Inf1 instance. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.

9.13.3 Arguments

- **func:** The `keras.Model` or function to be traced.
- **example_inputs:** A `tf.Tensor` or a tuple/list/dict of `tf.Tensor` objects for tracing the function. When `example_inputs` is a `tf.Tensor` or a list of `tf.Tensor` objects, we expect `func` to have calling signature `func(example_inputs)`. Otherwise, the expectation is that inference on `func` is done by calling `func(*example_inputs)` when `example_inputs` is a `tuple`, or `func(**example_inputs)` when `example_inputs` is a `dict`. The case where `func` accepts mixed positional and keyword arguments is currently unsupported.
- **subgraph_builder_function:** (Optional) A callable with signature

  `subgraph_builder_function(node : NodeDef) -> bool` (`NodeDef` is defined in tensorflow/core/framework/node_def.proto)

  that is used as a call-back function to determine which part of the tensorflow GraphDef given by tracing `func` will be placed on Machine Learning Accelerators.

  If `subgraph_builder_function` is not provided, then `trace` will automatically place operations on Machine Learning Accelerators or on CPU to maximize the execution efficiency.

  If it is provided, and `subgraph_builder_function(node)` returns `True`, and placing `node` on Machine Learning Accelerators will not cause deadlocks during execution, then `trace` will place `node` on Machine Learning Accelerators. If `subgraph_builder_function(node)` returns `False`, then `trace` will place `node` on CPU.

9.13.4 Special Flags

These are flags that get passed directly to the Neuron tracing API (rather than the Neuron Compiler). The flags are still passed via the environment variable `NEURON_CC_FLAGS`.

- **workdir:** example usage - `NEURON_CC_FLAGS='--workdir ./artifacts'` will create a folder named artifacts in the current directory and save artifacts that can be used for debug.
- **dynamic-batch-size:** example usage - `NEURON_CC_FLAGS='--dynamic-batch-size'` A flag to allow Neuron graphs to consume variable sized batches of data. Dynamic sizing is restricted to the 0th dimension of a tensor.
- **extract-weights (EXPERIMENTAL):** example usage - `NEURON_CC_FLAGS='--extract-weights'` will reduce the compiled model's protobuf size by taking the weights out of the protobuf. Useful for compiling large models that would exceed the 2GB protobuf size limit. This feature is experimental. Model performance is not guaranteed and the flag does not work in combination with `--neuroncore-pipeline-cores`, `--dynamic-batch-size`, and models with multiple NEFFs. Compiles models for the inf1.2xlarge instance.

9.13.5 Returns

- An AWS-Neuron-optimized `keras.Model`.

9.13.6 Example Usage

```
import tensorflow as tf
import tensorflow.neuron as tfn

input0 = tf.keras.layers.Input(3)
dense0 = tf.keras.layers.Dense(3)(input0)
model = tf.keras.Model(inputs=[input0], outputs=[dense0])
example_inputs = tf.random.uniform([1, 3])
model_neuron = tfn.trace(model, example_inputs)  # trace

model_dir = './model_neuron'
model_neuron.save(model_dir)
model_neuron_reloaded = tf.keras.models.load_model(model_dir)
```

9.13.7 Example Usage with Manual Device Placement Using *subgraph_builder_function*

```
import tensorflow as tf
import tensorflow.neuron as tfn

input0 = tf.keras.layers.Input(3)
dense0 = tf.keras.layers.Dense(3)(input0)
reshape0 = tf.keras.layers.Reshape([1, 3])(dense0)
output0 = tf.keras.layers.Dense(2)(reshape0)
model = tf.keras.Model(inputs=[input0], outputs=[output0])
example_inputs = tf.random.uniform([1, 3])

def subgraph_builder_function(node):
    return node.op == 'MatMul'
```

(continues on next page)

(continued from previous page)

```
model_neuron = tfn.trace(
    model, example_inputs,
    subgraph_builder_function=subgraph_builder_function,
)
```

Important: Although the old API `tensorflow.neuron.saved_model.compile` is still available under tensorflow-neuron 2.x, it supports only the limited capabilities of `tensorflow.neuron.trace` and will be deprecated in future releases.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.14 TensorFlow 1.x (`tensorflow-neuron`) Compilation API

The Neuron compilation API for TensorFlow 1.x enables compilation of saved model to an Inferentia target.

9.14.1 Method

`tensorflow.neuron.saved_model.compile`

9.14.2 Description

Within the graph or subgraph, the compile method selects and send Neuron-supported operations to Neuron-Compiler for compilation and saves the compiled artifacts in the graph. Uncompilable operations are kept as original operations for framework execution.

The compiled graph can be exported to saved model and served using TensorFlow Serving. Please see tensorflow-serving for more information about exporting to saved model and serving using TensorFlow Serving.

Options can be passed to Neuron compiler via the compile function. For example, the "`--neuroncore-pipeline-cores`" option directs Neuron compiler to compile each subgraph to fit in the specified number of NeuronCores. This number can be less than the total available NeuronCores on an Inf1 instance. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.

9.14.3 Arguments

- **model_dir:** The path of the original `SavedModel`.
- **new_model_dir:** The path to which the Neuron-optimized `SavedModel` will be stored.
- **batch_size:** (Optional) Positive integer representing batch size used in inference. The default value is 1.
- **model_shape_feed_dict:** (Optional) Dictionary {str: list} used for inferring tensor shapes. Keys should match model input names. Values are lists of positive integers representing model input tensor shapes.
- **model_feed_dict:** (Optional) Dictionary {str: numpy.array} used for inference. Useful for inferring tensor shapes. Keys should match model input names. Values are numpy arrays that can be fed as inputs to the `SavedModel`.

- **tags:** (Optional) Iterable of strings to identify the required `MetaGraphDef`. These should correspond to the tags used when saving the variables using the `SavedModel save()` API. Default is to use the first `tag_set` available in the `SavedModel`.
- **signature_def_key:** (Optional) String specifying the `signature_def` to use. Default is to use 'serving_default' or the first `signature_def` corresponding to `tags`.
- **minimum_segment_size:** (Optional) Integer indicating the minimum number of operations in an NeuronOp.
- **no_fuse_ops:** (Optional) None or iterable of strings (unordered) representing names of operations that are forcibly placed on CPU.
- **compiler_args:** (Optional) List of strings representing neuron-cc compiler arguments. Note that these arguments apply to all subgraphs generated by whitelist partitioning. For example, use `compiler_args=['--neuroncore-pipeline-cores', '4']` to set number of NeuronCores per subgraph to 4. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.
- **compiler_workdir:** (Optional) String representing work directory of the neuron-cc compiler.

9.14.4 Returns

- Dictionary with operator counts before/after optimization.
- Operator count statistics are displayed to show original count, post-optimization count, and the number placed on Neuron runtime. For example:

```
INFO:tensorflow:Number of operations in TensorFlow session: 3978
INFO:tensorflow:Number of operations after tf.neuron optimizations: 555
INFO:tensorflow:Number of operations placed on Neuron runtime: 554
```

9.14.5 Example Usage

```
import shutil
import tensorflow.neuron as tfn
saved_model_path = "<saved model path>"
compiled_saved_model_path = "<compiled saved model path>"
shutil.rmtree(compiled_saved_model_path, ignore_errors=True)
tfn.saved_model.compile(saved_model_path, compiled_saved_model_path)
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.15 TensorFlow Neuron (`tensorflow-nueron`) Auto Multicore Replication (Experimental)

The Neuron auto multicore replication Python API enables modifying TensorFlow 2.x traced models so that they can be automatically replicated across multiple cores. For Tensorflow-Serving models and TensorFlow 1.x models, see *TensorFlow Neuron TF1.x/TF2.x (tensorflow-nueron TF1.x/TF2.x) Auto Multicore Replication CLI (Experimental)*

Table of contents

- *TensorFlow Neuron TF 2.x (`tensorflow-nueron TF2.x`) Auto Multicore Replication Python API (Experimental)*
- *TensorFlow Neuron TF1.x/TF2.x (`tensorflow-nueron TF1.x/TF2.x`) Auto Multicore Replication CLI (Experimental)*

9.15.1 TensorFlow Neuron TF 2.x (`tensorflow-nueron TF2.x`) Auto Multicore Replication Python API (Experimental)

Method

`tensorflow.neuron.auto_multicore`

Description

Converts an existing AWS-Neuron-optimized `keras.Model` and return an auto-replication tagged AWS-Multicore-Neuron-optimized `keras.Model` that can execute on AWS Machine Learning Accelerators. Like the traced model, the returned `keras.Model` will support inference only. Attributes or variables held by the original function or `keras.Model` will be dropped.

The auto model replication feature in TensorFlow-Neuron enables you to create a model once and the model parallel replication would happen automatically. The desired number of cores can be less than the total available NeuronCores on an Inf1 instance but not less than 1. This reduces framework memory usage as you are not loading the same model multiple times manually. Calls to the returned model will execute the call on each core in a round-robin fashion.

The returned `keras.Model` can be exported as SavedModel and served using TensorFlow Serving. Please see tensorflow-serving for more information about exporting to saved model and serving using TensorFlow Serving.

Note that the automatic replication will only work on models compiled with pipeline size 1: via `--neuroncore-pipeline-cores=1`. If auto replication is not enabled, the model will default to replicate on up to 4 cores.

See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.

Arguments

- **func:** The `keras.Model` or function to be traced.
- **example_inputs:** A `tf.Tensor` or a tuple/list/dict of `tf.Tensor` objects for tracing the function. When `example_inputs` is a `tf.Tensor` or a list of `tf.Tensor` objects, we expect `func` to have calling signature `func(example_inputs)`. Otherwise, the expectation is that inference on `func` is done by calling `func(*example_inputs)` when `example_inputs` is a `tuple`, or `func(**example_inputs)` when `example_inputs` is a `dict`. The case where `func` accepts mixed positional and keyword arguments is currently unsupported.
- **num_cores:** The desired number of cores where the model will be automatically replicated across

Returns

- An AWS-Multicore-Neuron-optimized `keras.Model`.

Example Python API Usage for TF2.x traced models:

```
input0 = tf.keras.layers.Input(3)
dense0 = tf.keras.layers.Dense(3)(input0)
inputs = [input0]
outputs = [dense0]
model = tf.keras.Model(inputs=inputs, outputs=outputs)
input0_tensor = tf.random.uniform([1, 3])
model_neuron = tfn.trace(model, input0_tensor)

num_cores = 4
multicore_model = tfn.auto_multicore(model_neuron, input0_tensor, num_cores=num_cores)
multicore_model(input0_tensor)
```

Example Python API Usage for TF2.x saved models:

```
from tensorflow.python import saved_model

input0_tensor = tf.random.uniform([1, 3])
num_cores = 4
reload_model = saved_model.load(model_dir)
multicore_model = tfn.auto_multicore(reload_model, input0_tensor, num_cores=num_cores)
```

9.15.2 TensorFlow Neuron TF1.x/TF2.x (`tensorflow-nueron TF1.x/TF2.x`) Auto Multicore Replication CLI (Experimental)

The Neuron auto multicore replication CLI enables modifying TensorFlow 1.x and Tensorflow 2.x traced saved models so that they can be automatically replicated across multiple cores. By performing this call on Tensorflow Saved Models, we can support both Tensorflow-Serving and Tensorflow 1.x without significant modifications to the code. Note that the python API does not support Tensorflow 1.x.

Method

```
tf-neuron-auto-multicore MODEL_DIR --num_cores NUM_CORES --new_model_dir NEW_MODEL_DIR
```

Arguments

- **MODEL_DIR:** The directory of a saved AWS-Neuron-optimized `keras.Model`.
- **NUM_CORES:** The desired number of cores where the model will be automatically replicated across
- **NEW_MODEL_DIR:** The directory of where the AWS-Multicore-Neuron-optimized `keras.Model` will be saved

Example CLI Usage for TF 1.x and Tensorflow-Serving saved models:

```
tf-neuron-auto-multicore ./resnet --num_cores 8 --new_model_dir ./modified_resnet
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.16 TensorFlow Neuron (`tensorflow-neuron (TF1.x)`) Release Notes

Table of contents

- *Known Issues and Limitations - updated 08/12/2021*
- *tensorflow-neuron 1.x release [2.4.0.0]*
- *tensorflow-neuron 1.x release [2.3.0.0]*
- *tensorflow-neuron 1.x release [2.1.14.0]*
- *tensorflow-neuron 1.x release [2.1.14.0]*
- *tensorflow-neuron 1.x release [2.1.13.0]*
- *tensorflow-neuron 1.x release [2.1.6.0]*
- *tensorflow-neuron 1.x release [2.0.4.0]*
- *tensorflow-neuron 1.x release [2.0.3.0]*
- *[1.15.5.1.5.1.0]*
- *[1.15.5.1.4.0.0]*
- *[1.15.5.1.3.3.0]*
- *[1.15.5.1.2.9.0]*
- *[1.15.5.1.2.8.0]*
- *[1.15.5.1.2.2.0]*
- *[1.15.4.1.1.3.0]*
- *[1.15.4.1.0.2168.0]*
- *[1.15.3.1.0.2043.0]*
- *[1.15.3.1.0.1965.0]*
- *[1.15.3.1.0.1953.0]*
- *[1.15.3.1.0.1891.0]*
- *[1.15.2.1.0.1796.0]*
- *[1.15.2.1.0.1572.0]*
- *[1.15.0.1.0.1333.0]*
- *[1.15.0.1.0.1240.0]*
- *[1.15.0.1.0.997.0]*

- *[1.15.0.1.0.803.0]*
- *[1.15.0.1.0.749.0]*
- *[1.15.0.1.0.663.0]*

This document lists the release notes for the tensorflow-neuron 1.x package.

9.16.1 Known Issues and Limitations - updated 08/12/2021

- Support on serialized TensorFlow 2.x custom operators is currently limited. Serializing some operators registered from TensorFlow-text through TensorFlow Hub is going to cause failure in tensorflow.neuron.trace.
- Issue: When compiling large models, user might run out of memory and encounter this fatal error.

```
terminate called after throwing an instance of 'std::bad_alloc'
```

Solution: run compilation on a c5.4xlarge instance type or larger.

- Issue: When upgrading `tensorflow-neuron` with `pip install tensorflow-neuron --upgrade`, the following error message may appear, which is caused by `pip` version being too low.

```
Could not find a version that satisfies the requirement TensorFlow<1.16.0,>=1.15.0 (from␣
↪tensorflow-neuron)
```

Solution: run a `pip install pip --upgrade` before upgrading `tensorflow-neuron`.

- Issue: Some Keras routines throws the following error:

```
AttributeError: 'str' object has no attribute 'decode'.
```

Solution: Please downgrade *h5py* by *pip install 'h5py<3'*. This is caused by https://github.com/TensorFlow/TensorFlow/issues/44467.

9.16.2 tensorflow-neuron 1.x release [2.4.0.0]

Date: 11/23/2022

- Introduce `tf-neuron-auto-multicore` tool to enable automatic data parallel on multiple NeuronCores.
- Deprecated the NEURONCORE_GROUP_SIZES environment variable.
- Minor bug fixes.

9.16.3 tensorflow-neuron 1.x release [2.3.0.0]

Date: 04/29/2022

- Minor bug fixes.

9.16.4 tensorflow-neuron 1.x release [2.1.14.0]

Date: 03/25/2022

- Minor bug fixes.

9.16.5 tensorflow-neuron 1.x release [2.1.14.0]

Date: 02/17/2022

- Minor bug fixes.

9.16.6 tensorflow-neuron 1.x release [2.1.13.0]

Date: 02/16/2022

- Fixed a bug that caused a memory leak. The memory leak was approximately 128b for each inference and exists in all versions of TensorFlow Neuron versions part of Neuron 1.16.0 to Neuron 1.17.0 releases. see pre-release-content for exact versions included in each release.

9.16.7 tensorflow-neuron 1.x release [2.1.6.0]

Date: 01/20/2022

- Enhanced auto data parallel (e.g. when using NEURONCORE_GROUP_SIZES=X,Y,Z,W) to support edge cases.
- Added new operators support. see *TensorFlow Neuron (tensorflow-neuron (TF1.x)) Supported operators*.

9.16.8 tensorflow-neuron 1.x release [2.0.4.0]

Date: 11/05/2021

- Updated Neuron Runtime (which is integrated within this package) to `libnrt 2.2.18.0` to fix a container issue that was preventing the use of containers when /dev/neuron0 was not present. See details here neuron-runtime-release-notes.

9.16.9 tensorflow-neuron 1.x release [2.0.3.0]

Date: 10/27/2021

New in this release

- TensorFlow 1.x (`tensorflow-neuron`) now support Neuron Runtime 2.x (`libnrt.so` shared library) only.

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.

- Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

Resolved Issues

- Fix neuron-cc argument handling bug when nothing can be compiled.
- Fixing the support of cast operators applied after constants, by Introducing support of constant-folding pass before Neuron auto-mixed-precision.

9.16.10 [1.15.5.1.5.1.0]

Date: 07/02/2021

New in this release

- Bug fixes regarding scalar inputs/outputs.
- Minor performance improvements when dynamic batch size is turned on or when model is small.

9.16.11 [1.15.5.1.4.0.0]

Date: 05/28/2021

New in this release

- Reduce the amount of input/output data movement during inference.
- Improve parallelism for dynamic batch size inference by adopting a new sharding mechanism.
- Reduce the amount of host memory usage during inference.
- tfn.saved_model.compile now generates correct code when operator Split is used as output.
- tfn.saved_model.compile now properly reads input tensor shape information from SignatureDef proto.
- tfn.saved_model.compile now terminates properly when neuron-cc compiler argument is passed but there is no successful compilation.
- Fix bug on some wrong internal tensor names when neuron-cc compiler crashes.
- Other minor bug fixes.

9.16.12 [1.15.5.1.3.3.0]

Date: 05/01/2021

New in this release

1. Minor enhancements.

9.16.13 [1.15.5.1.2.9.0]

Date: 03/04/2021

New in this release

1. Minor enhancements.

9.16.14 [1.15.5.1.2.8.0]

Date: 02/24/2021

New in this release

1. Fix for CVE-2021-3177.

9.16.15 [1.15.5.1.2.2.0]

Date: 01/30/2021

New in this release

1. Bug fixes and internal refactor.
2. Bump TensorFlow base package version to 1.15.5.
3. Introduced a new argument `convert_constants_to_variables` to the compilation API `tfn.saved_model.compile`. Setting it to `True` can address the issue of large constants consuming too much memory in the TensorFlow runtime.

9.16.16 [1.15.4.1.1.3.0]

Date: 12/23/2020

New in this release

1. Improved logging during *tfn.saved_model.compile* to display *neuron-cc* compilation progress.
2. Small performance improvement in some edge cases by optimizing the NeuronCore-executable assignment mechanism.

9.16.17 [1.15.4.1.0.2168.0]

Date: 11/17/2020

New in this release

1. tensorflow-neuron is now a plugin package that can be used together with TensorFlow~=1.15.0 built with `GLIBCXX_USE_CXX11_ABI=0`.
2. Improved logging during `tfn.saved_model.compile` to display `neuron-cc` logging file path, which is useful for tracking `neuron-cc` compilation progress.
3. Small performance improvement by utilizing shared memory more efficiently.

9.16.18 [1.15.3.1.0.2043.0]

Date: 09/22/2020

New in this release

1. tensorflow-neuron now automatically enables data parallel mode on four cores in one Inferentia. In `TensorFlow-model-server-neuron`, most models can now fully utilize four cores automatically. In Python TensorFlow, running threaded inference using `>=4` Python threads in the same TensorFlow Session lead to full utilization of four cores.
2. tensorflow-neuron now tries to enable dynamic batch size automatically for a limited number of models, such as ResNet50.
3. Improved logging during `tfn.saved_model.compile` to display input/output information about subgraphs that are going to be compiled by `neuron-cc`.

9.16.19 [1.15.3.1.0.1965.0]

Date: 08/08/2020

New in this release

Various minor improvements.

9.16.20 [1.15.3.1.0.1953.0]

Date: 08/05/2020

New in this release

Various minor improvements.

9.16.21 [1.15.3.1.0.1891.0]

Date: 07/16/2020

New in this release

This version contains a few bug fixes and user experience improvements.

Dependency change

1. Bump TensorFlow base package version number to 1.15.3
2. Add `TensorFlow >= 1.15.0, < 1.16.0` as an installation dependency so that packages depending on TensorFlow can be installed together with tensorflow-neuron without error

New Features

1. `tensorflow-neuron` now displays a summary of model performance when profiling is enable by setting environment variable `NEURON_PROFILE`

Resolved Issues

1. Environment variable `NEURON_PROFILE` can now be set to a non-existing path which will be automatically created
2. Fixed a bug in `tfn.saved_model.compile` that causes compilation failure when `dynamic_batch_size=True` is specified on a SavedModel with unknown rank inputs.

9.16.22 [1.15.2.1.0.1796.0]

Date 6/11/2020

New in this release

This version contains a few bug fixes.

Major New Features

Resolved Issues

1. Fixed a bug related with device placement. Now models with device information hardcoded to GPU can be successfully compiled with `tfn.saved_model.compile`
2. Fixed a bug in `tfn.saved_model.compile` that causes models containing Reshape operators not functioning correctly when it is compiled with `dynamic_batch_size=True`
3. Fixed a bug in `tfn.saved_model.compile` that causes models containing Table related operators to initialize incorrectly after compilation.

Known Issues and limitations

9.16.23 [1.15.2.1.0.1572.0]

Date: 5/11/2020

New in this release

This version contains some bug fixes and new features.

Major New Features

- tensorflow-neuron is now built on TensorFlow 1.15.2 instead of TensorFlow 1.15.0

Resolved Issues

- Fixed a bug that caused Neuron runtime resources to not all be released when a tensorflow-neuron process terminated with in-flight inferences
- Inference timeout value set at compile time is now correctly recognized at runtime

Known Issues and limitations

9.16.24 [1.15.0.1.0.1333.0]

Date: 3/26/2020

New in this release

Major New Features

- Improved performance between TensorFlow to Neuron runtime.

Resolved Issues

- Fixed a bug in Neuron runtime adaptor operator's shape function when dynamic batch size inference is enabled
- Framework method (tensorflow.neuron.saved-model.compile) improved handling of compiler timeout termination by letting it clean up before exiting.

Known Issues and limitations

9.16.25 [1.15.0.1.0.1240.0]

Date: 2/27/2020

New in this release

Major New Features

- Enabled runtime memory optimizations by default to improve inference performance, specifically in cases with large input/output tensors
- tfn.saved_model.compile now displays warning message instead of "successfully compiled" if less than 30% of operators are mapped to Inferentia
- Improve error messages. Runtime failure error messages are now more descriptive and also provide instructions to restart neuron-rtd when necessary.

Resolved Issues

Known Issues and Limitations

- Issue: When compiling a large model, may encounter.

```
terminate called after throwing an instance of 'std::bad_alloc'
```

Solution: run compilation on c5.4xlarge instance type or larger.

Other Notes

9.16.26 [1.15.0.1.0.997.0]

Date: 1/27/2020

New in this release

Major New Features

- Added support for NCHW pooling operators in tfn.saved_model.compile.

Resolved Issues

- Fixed GRPC transient status error issue.
- Fixed a graph partitioner issue with control inputs.

Known Issues and Limitations

- Issue: When compiling a large model, may encounter.

```
terminate called after throwing an instance of 'std::bad_alloc'
```

Solution: run compilation on c5.4xlarge instance type or larger.

Other Notes

9.16.27 [1.15.0.1.0.803.0]

Date: 12/20/2019

New in this release

Major New Features

Resolved Issues

- Improved handling of `tf.neuron.saved_model.compile` arguments

Known Issues and Limitations

Other Notes

9.16.28 [1.15.0.1.0.749.0]

Date: 12/1/2019

New in this release

Major New Features

Resolved Issues

- Fix race condition between model load and model unload when the process is killed
- Remove unnecessary GRPC calls when the process is killed

Known Issues and Limitations

- When compiling a large model, may encounter "terminate called after throwing an instance of 'std::bad_alloc'". Solution: run compilation on c5.4xlarge instance type or larger.
- The pip package `wrapt` may have a conflicting version in some installations. This is seen when this error occurs:

```
ERROR: Cannot uninstall 'wrapt'. It is a distutils installed project and thus we cannot␣
↪accurately determine which files belong to it which would lead to only a partial␣
↪uninstall.
```

To solve this, you can update wrapt to the newer version:

```
python3 -m pip install wrapt --ignore-installed
python3 -m pip install tensorflow-neuron
```

Within a Conda environment:

```
conda update wrapt
conda update tensorflow-neuron
```

Other Notes

9.16.29 [1.15.0.1.0.663.0]

Date: 11/25/2019

New in this release

This version is available only in released DLAMI v26.0 and is based on TensorFlow version 1.15.0. Please update to latest version.

Major New Features

Resolved Issues

Known Issues and Limits

Models Supported

The following models have successfully run on neuron-inferentia systems

1. BERT_LARGE and BERT_BASE
2. Transformer
3. Resnet50 V1/V2
4. Inception-V2/V3/V4

Other Notes

- Python versions supported:
  - 3.5, 3.6, 3.7
- Linux distribution supported:
  - Ubuntu 18, Amazon Linux 2

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.17 TensorFlow Neuron (`tensorflow-neuron (TF2.x)`) Release Notes

Table of contents

- *Known Issues and Limitations - updated 08/12/2021*
- *tensorflow-neuron 2.x release [2.4.0.0]*
- *tensorflow-neuron 2.x release [2.3.0.0]*
- *tensorflow-neuron 2.x release [2.2.0.0]*
- *tensorflow-neuron 2.x release [2.1.14.0]*
- *tensorflow-neuron 2.x release [2.1.13.0]*
- *tensorflow-neuron 2.x release [2.1.6.0]*
- *tensorflow-neuron 2.x release [2.0.4.0]*
- *tensorflow-neuron 2.x release [2.0.3.0]*
- *tensorflow-neuron 2.x release [1.6.8.0]*

This document lists the release notes for the tensorflow-neuron 2.x packages.

9.17.1 Known Issues and Limitations - updated 08/12/2021

- Support on serialized TensorFlow 2.x custom operators is currently limited. Serializing some operators registered from tensorflow-text through TensorFlow Hub is going to cause failure in tensorflow.neuron.trace.
- Memory leak exists on latest releases of TensorFlow Neuron for versions 2.1, 2.2, 2.3, and 2.4.
- Issue: When compiling large models, user might run out of memory and encounter this fatal error.

```
terminate called after throwing an instance of 'std::bad_alloc'
```

Solution: run compilation on a c5.4xlarge instance type or larger.

- Issue: When upgrading `tensorflow-neuron` with `pip install tensorflow-neuron --upgrade`, the following error message may appear, which is caused by `pip` version being too low.

```
Could not find a version that satisfies the requirement tensorflow<1.16.0,>=1.15.0 (from␣
↪tensorflow-neuron)
```

Solution: run a `pip install pip --upgrade` before upgrading `tensorflow-neuron`.

- Issue: Some Keras routines throws the following error:

```
AttributeError: 'str' object has no attribute 'decode'.
```

Solution: Please downgrade *h5py* by *pip install 'h5py<3'*. This is caused by https://github.com/TensorFlow/TensorFlow/issues/44467.

9.17.2 tensorflow-neuron 2.x release [2.4.0.0]

Date: 11/22/2022

- Experimental support for tracing models larger than 2 GB through environment variable `NEURON_CC_FLAGS='--extract-weights'`.
- Introduce `tfn.auto_multicore` Python API to enable automatic data parallel on multiple NeuronCores.
- Introduce `tf-neuron-auto-multicore` tool to enable automatic data parallel on multiple NeuronCores.
- Deprecated the NEURONCORE_GROUP_SIZES environment variable.
- Minor bug fixes.

9.17.3 tensorflow-neuron 2.x release [2.3.0.0]

Date: 04/29/2022

- Added support for Tensorflow 2.8.0.
- Added support for Slice operator
- The graph partitioner now prefers to place less compute intensive operators on CPU if the model already contains a large amount of compute intensive operators.
- Fixed Github issue #408, the fix solves data type handling bug in `tfn.trace` when the model contains Conv2D operators.

9.17.4 tensorflow-neuron 2.x release [2.2.0.0]

Date: 03/25/2022

- Updated TensorFlow 2.5 to version 2.5.3.
- Added support for TensorFlow 2.6 and 2.7.
- Added a warning message when calling `tfn.saved_model.compile` API. In tensorflow-neuron 2.x you should call *tensorflow.neuron.trace*. `tfn.saved_model.compile` API supports only partial functionality of *tensorflow.neuron.trace* and will be deprecated in the future.

9.17.5 tensorflow-neuron 2.x release [2.1.14.0]

Date: 02/17/2022

- Fixed a bug in TensorFlow Neuron versions 2.1, 2.2. 2.3 and 2.4. The fixed bug was causing a memory leak of 128 bytes for each inference.
- Improved warning message when calling deprecated compilation API under tensorflow-neuron 2.x.

9.17.6 tensorflow-neuron 2.x release [2.1.13.0]

Date: 02/16/2022

- Fixed a bug that caused a memory leak. The memory leak was approximately 128b for each inference and exists in all versions of TensorFlow Neuron versions part of Neuron 1.16.0 to Neuron 1.17.0 releases. see pre-release-content for exact versions included in each release. This release only addresses the leak in TensorFlow Neuron 2.5. Future release of TensorFlow Neuron will fix the leak in other versions as well (2.1, 2.2, 2.3, 2.4).

9.17.7 tensorflow-neuron 2.x release [2.1.6.0]

Date: 01/20/2022

- Updated TensorFlow 2.5 to version 2.5.2.
- Enhanced auto data parallel (e.g. when using NEURONCORE_GROUP_SIZES=X,Y,Z,W) to support edge cases.
- Fixed a bug that may cause tensorflow-neuron to generate in some cases scalar gather instruction with incorrect arguments.

9.17.8 tensorflow-neuron 2.x release [2.0.4.0]

Date: 11/05/2021

- Updated Neuron Runtime (which is integrated within this package) to `libnrt 2.2.18.0` to fix a container issue that was preventing the use of containers when /dev/neuron0 was not present. See details here neuron-runtime-release-notes.

9.17.9 tensorflow-neuron 2.x release [2.0.3.0]

Date: 10/27/2021

New in this release

- TensorFlow 2.x (`tensorflow-neuron`) now support Neuron Runtime 2.x (`libnrt.so` shared library) only.

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.

  - Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

- Updated TensorFlow 2.3.x from TensorFlow 2.3.3 to TensorFlow 2.3.4.
- Updated TensorFlow 2.4.x from TensorFlow 2.4.2 to TensorFlow 2.4.3.
- Updated TensorFlow 2.5.x from TensorFlow 2.5.0 to TensorFlow 2.5.1.

Resolved Issues

- Fix bug that can cause illegal compiler optimizations
- Fix bug that can cause dynamic-shape operators be placed on Neuron

9.17.10 tensorflow-neuron 2.x release [1.6.8.0]

Date: 08/12/2021

New in this release

- First release of TensorFlow 2.x integration, Neuron support now TensorFlow versions 2.1.4, 2.2.3, 2.3.3, 2.4.2, and 2.5.0.
- New public API tensorflow.neuron.trace: trace a TensorFlow 2.x keras.Model or a Python callable that can be decorated by tf.function, and return an AWS-Neuron-optimized keras.Model that can execute on AWS Machine Learning Accelerators.

  Please note that TensorFlow 1.x SavedModel compilation API tensorflow.neuron.saved_model.compile is not supported in tensorflow-neuron 2.x . It continues to function in tensorflow-neuron 1.15.x .
- Included versions:
  - tensorflow-neuron-2.5.0.1.6.8.0
  - tensorflow-neuron-2.4.2.1.6.8.0
  - tensorflow-neuron-2.3.3.1.6.8.0
  - tensorflow-neuron-2.2.3.1.6.8.0
  - tensorflow-neuron-2.1.4.1.6.8.0

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.18 TensorFlow Neuron (`tensorflow-neuron (TF2.x)`) Accelerated (torch-neuron) Python APIs and Graph Ops

This page lists TensorFlow 2.x Python APIs and graph operators that are accelerated by AWS Neuron. The lists are not exhaustive. TensorFlow 2.x Python APIs or graph operators that are not listed here may still be accelerated if they are composed of accelerated primitives, or they will be executed on CPU without significant acceleration. The TensorFlow Neuron integration contains an automatic operator-device-placement mechanism that strives to maximize the execution efficiency of your deep learning models on AWS Machine Learning ASIC instances.

9.18.1 Accelerated Python APIs

Module	Accelerated Python API	Comments
tf	tf.abs	
	tf.add	
	tf.add_n	
	tf.broadcast_static_shape	
	tf.cast	
	tf.constant	
	tf.convert_to_tensor	
	tf.cumsum	axis must be a compile-time constant.
	tf.einsum	
	tf.erf	
	tf.exp	
	tf.identity	
	tf.matmul	Uses float16/bfloat16 matmul with float32 accumulation.
	tf.maximum	
	tf.minimum	
	tf.multiply	
	tf.negative	
	tf.range	start, limit and delta arguments must be compile-time constants.
	tf.realdiv	
	tf.reciprocal	
	tf.reduce_all	axis must be a compile-time constant.
	tf.reduce_any	axis must be a compile-time constant.
	tf.reduce_max	axis must be a compile-time constant.
	tf.reduce_min	axis must be a compile-time constant.
	tf.reduce_prod	axis must be a compile-time constant.
	tf.reduce_sum	axis must be a compile-time constant.
	tf.reshape	shape argument must be a compile-time constant.
	tf.rsqrt	
	tf.scalar_mul	
	tf.shape	
	tf.shape_n	
	tf.sigmoid	
	tf.size	

continues on next page

Table 1 – continued from previous page

Module	Accelerated Python API	Comments
	tf.slice	size must be a compile-time constant. In addition, either begin must be a compile-time constant or size must be non-negative.
	tf.sqrt	
	tf.square	
	tf.squared_difference	
	tf.squeeze	
	tf.stack	
	tf.stop_gradient	
	tf.strided_slice	
	tf.tanh	
	tf.tensordot	
	tf.to_bfloat16	
	tf.to_float	
	tf.truediv	
tf.layers	tf.layers.batch_normalization	
	tf.layers.dense	
	tf.layers.flatten	
tf.nn	tf.nn.batch_normalization	
	tf.nn.bias_add	
	tf.nn.dropout	Always treated as tf.identity during inference.
	tf.nn.fused_batch_norm	
	tf.nn.leaky_relu	
	tf.nn.relu	
	tf.nn.relu6	
	tf.nn.relu_layer	
	tf.nn.softmax	

9.18.2 Accelerated graph operators

```
Add
AddN
AddV2
BatchMatMul
BatchMatMulV2
BiasAdd
Cast
Const
Cumsum
Einsum
Erf
Exp
ExpandDims
FusedBatchNorm
FusedBatchNormV2
```

(continues on next page)

(continued from previous page)

```
FusedBatchNormV3
Greater
Identity
LeakyRelu
MatMul
Max
Maximum
Minimum
Mean
Mul
Neg
Pack
RealDiv
Relu
Relu6
Reshape
Rsqrt
Sigmoid
Softmax
Split
SplitV
Sqrt
Square
SquaredDifference
Squeeze
StridedSlice
Sub
Sum
Tanh
Transpose
Unpack
```

The lists share many commonalities with Available TensorFlow Ops. Portions of this page are modifications based on work created and shared by Google and used according to terms described in the Creative Commons 4.0 Attribution License.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

9.19 TensorFlow Neuron (`tensorflow-neuron (TF1.x)`) Supported operators

To see a list of supported operators for TensorFlow 1.x, run the following command:

`neuron-cc list-operators --framework TENSORFLOW`

9.19.1 Neuron Compiler Release [1.9.1.0]

Date: 01/20/2022

Added

```
isNan
FusedBatchNormV3
```

9.19.2 Neuron Compiler Release [1.7.3.0]

Added

```
ArgMax
ArgMin
```

9.19.3 Neuron Compiler Release [1.6.13.0]

No changes

9.19.4 Neuron Compiler Release [1.5.5.0]

No changes

9.19.5 Neuron Compiler Release [1.4.0.0]

No changes

9.19.6 Neuron Compiler Release [1.3.0.0]

Added

```
Abs
Cos
DepthwiseConv2dNative
Erf
Rank
Sin
Size
```

9.19.7 Neuron Compiler Release [1.2.7.0]

No changes

9.19.8 Neuron Compiler Release [1.2.2.0]

Added

```
AdjustContrastv2
AdjustSaturation
BroadcastTo
Cholesky
Conv2DBackpropInput
Conv3D
CropAndResize
FloorDiv
HSVToRGB
InvertPermutation
L2Loss
Log1p
MatrixBandPart
MatrixDiag
MatrixSetDiag
MatrixTriangularSolve
MaxPool3D
MirrorPad
RGBToHSV
Range
SoftmaxCrossEntropyWithLogits
SquaredDifference
StopGradient
Unpack
UnsortedSegmentSum
```

9.19.9 Neuron Compiler Release [1.0.24045.0]

Added `FloorDiv`, `Softplus`, `Unstack`

9.19.10 Neuron Compiler Release [1.0.18001]

No changes

9.19.11 Neuron Compiler Release [1.0.16764]

Added:

```
LogSoftmax
Neg
ResizeBilinear
ResizeNearestNeighbor
```

9.19.12 Neuron Compiler Release [1.0.15275]

Added

```
Neg
```

Removed

```
Log
```

(was inadvertently advertised as supported)

9.19.13 Neuron Compiler Release [1.0.12696]

No changes

9.19.14 Neuron Compiler Release [1.0.9410]

No changes

9.19.15 Neuron Compiler Release [1.0.7878]

No changes

9.19.16 Neuron Compiler Release [1.0.6801]

No changes

9.19.17 Neuron Compiler Release [1.0.5939]

No changes

9.19.18 Neuron Compiler Release [1.0.5301]

No changes

9.19.19 Neuron Compiler Release [1.0.4680.0]

```
Add
AddV2
All
AvgPool
BatchMatMul
BatchMatMulV2
BatchToSpaceND
BiasAdd
Cast
Ceil
Concat
ConcatV2
Const
Conv2D
Equal
Exp
ExpandDims
Fill
Floor
FusedBatchNorm
Greater
GreaterEqual
Identity
LRN
LeakyRelu
Less
LessEqual
Log
LogicalAnd
LogicalNot
LogicalOr
MatMul
Max
MaxPool
Maximum
Mean
Min
Minimum
Mul
NoOp
NotEqual
Pack
Pad
PadV2
Placeholder
Pow
```

(continues on next page)

(continued from previous page)

```
Prod
RandomUniform
RealDiv
Reciprocal
Relu
Relu6
Reshape
ReverseV2
Round
Rsqrt
Select
Shape
Sigmoid
Sign
Slice
Softmax
SpaceToBatchND
Split
SplitV
Sqrt
Square
Squeeze
StridedSlice
Sub
Sum
Tanh
Tile
Transpose
ZerosLike
```

This document is relevant for: `Inf1`

Training

Note: TensorFlow Neuron support is coming soon.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`

CHAPTER

TEN

MXNET NEURON

MXNet Neuron unlocks high-performance and cost-effective deep learning acceleration on AWS Trainium-based and Inferentia-based Amazon EC2 instances.

MXNet Neuron enables native MXNet models to be accelerated on Neuron devices, so you can use your existing framework application and get started easily with minimal code changes.

Inference

Setup Guide for Inf1

This document is relevant for: `Inf1`

10.1 Install MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

10.1.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

MXNet 1.8.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install mx_neuron neuron-cc
```

MXNet 1.5.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
```

(continues on next page)

(continued from previous page)

```
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.* neuron-cc
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

10.1.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

MXNet 1.5.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.* neuron-cc
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

10.1.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

MXNet 1.8.0

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron
```

MXNet 1.5.1

Ubuntu 18 AMI/Ubuntu 20 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.*
```

Amazon Linux 2 AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.*
```

Ubuntu 18 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on Neuron documentation
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

Amazon Linux 2 DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.2 Update to latest MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

10.2.1 Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
```

(continues on next page)

(continued from previous page)

```
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Update Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Update Neuron Tools
sudo yum update aws-neuronx-tools -y

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

10.2.2 Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron neuron-cc
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.* neuron-cc
```

10.2.3 Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
```

(continues on next page)

(continued from previous page)

```
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mx_neuron
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.*
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum update aws-neuronx-dkms -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Activate a Python 3.7 virtual environment where Neuron pip packages were installed
```

(continues on next page)

(continued from previous page)

```
source mxnet_venv/bin/activate


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.*
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Update OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Update Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.*
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Update OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum update aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Update Neuron MXNet
pip install --upgrade mxnet_neuron==1.5.1.*
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3 Install previous MXNet Neuron releases

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

This document is relevant for: `Inf1`

10.3.1 Install MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.1.4.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.11.4.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0 neuron-cc==1.11.4.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
```

(continues on next page)

(continued from previous page)

```
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install mx_neuron==1.8.0.2.2.2.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
```

(continues on next page)

(continued from previous page)

```
pip install mxnet_neuron==1.5.1.1.10.0.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.3.3.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.3.3.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.10.0.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.2 Install MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#####################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.7900.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.7900.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.790.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
```

(continues on next page)

(continued from previous page)

```
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0 neuron-cc==1.10.3.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0 neuron-cc==1.10.3.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
```

(continues on next page)

(continued from previous page)

```
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-
↪manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.2.2.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.14.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.14.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.9.0.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.3 Install MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.623.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0 neuron-cc==1.9.1.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0 neuron-cc==1.9.1.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
```

(continues on next page)

(continued from previous page)

```
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
```

(continues on next page)

(continued from previous page)

```
pip install mx_neuron==1.8.0.2.1.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.1.5.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
```

(continues on next page)

(continued from previous page)

```
pip install mxnet_neuron==1.5.1.1.8.0.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.13.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.13.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.8.0.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.4 Install MXNet Neuron

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- For an example of how to install Neuron components in a container, see tutorial-docker-env-setup and our neuron-containers documentation for more details.

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
```

(continues on next page)

(continued from previous page)

```
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-2.0.494.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
```

(continues on next page)

(continued from previous page)

```
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0 neuron-cc==1.8.5.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0 neuron-cc==1.8.5.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
```

(continues on next page)

(continued from previous page)

```
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y
```

(continues on next page)

(continued from previous page)

```
# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.2.0.290.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB
```

(continues on next page)

(continued from previous page)

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.2.8.0 --allow-change-held-packages -y
```

(continues on next page)

(continued from previous page)

```
##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.2.8.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.7.3.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.5 Install MXNet Neuron (Neuron 1.15.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##############################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
```

(continues on next page)

(continued from previous page)

```
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.6 Install MXNet Neuron (Neuron 1.15.1)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.25.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
```

(continues on next page)

(continued from previous page)

```
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.1.5.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.1.5.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.1.5.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.24.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.7 Install MXNet Neuron (Neuron 1.15.0)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.20.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0 neuron-cc==1.6.13.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0 neuron-cc==1.6.13.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.4.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
```

(continues on next page)

(continued from previous page)

```
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.450.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.450.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.450.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.19.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.5.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.3.8 Install MXNet Neuron (Neuron 1.14.2)

Table of contents

- *Develop on AWS ML accelerator instance*
- *Compile on compute instance*
- *Deploy on AWS ML accelerator instance*

Develop on AWS ML accelerator instance

The simplest environment setup for model development installs all Neuron SDK components directly on an AWS ML accelerator instance: the Neuron framework extensions, compiler, runtime, and tools. This will allow you to compile, execute, and performance tune your model, all in the same instance. This is the recommended workflow when first starting to work with Neuron device or when optimizing a model.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
```

(continues on next page)

(continued from previous page)

```
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
```

(continues on next page)

(continued from previous page)

```
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
```

(continues on next page)

(continued from previous page)

```
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
```

(continues on next page)

(continued from previous page)

```
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.6 -m ipykernel install --user --name mxnet_venv --display-name "Python (Neuron␣
↪MXNet)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-tools=1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

# Install Neuron Tools
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-tools-1.7.10.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Compile on compute instance

If model compilation occurs outside the model deployment environment, you can install only the Neuron framework extensions and the compiler on any compute instance. This setup is helpful when compiling large complex models that require large amount of memory or during a CICD process where models are compiled in a separate step, prior to deployment.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
```

(continues on next page)

(continued from previous page)

```
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0 neuron-cc==1.5.5.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0 neuron-cc==1.5.5.0
```

Deploy on AWS ML accelerator instance

During deployment it can be beneficial to reduce the number of components installed in the system. For use-cases where only inference is necessary (compilation is already complete), only the framework and runtime should be installed.

MXNet 1.8.0

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.6
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##########################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##########################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
```

(continues on next page)

(continued from previous page)

```
#Install Neuron MXNet
wget https://aws-mx-pypi.s3-us-west-2.amazonaws.com/1.8.0/aws_mx_cu110-1.8.0-py2.py3-
↪none-manylinux2014_x86_64.whl
pip install aws_mx_cu110-1.8.0-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron==1.8.0.1.3.0.0
```

MXNet 1.5.1

Ubuntu AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
. /etc/os-release
sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.PUB |␣
↪sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.6
#
# sudo add-apt-repository ppa:deadsnakes/ppa
```

(continues on next page)

(continued from previous page)

```
# sudo apt-get install python3.6
#
#####################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.6-venv g++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0
```

Amazon Linux AMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-dkms-2.0.386.0 -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################
```

(continues on next page)

(continued from previous page)

```
# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.6-venv gcc-c++
python3.6 -m venv mxnet_venv
source mxnet_venv/bin/activate
pip install -U pip


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0
```

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo apt-get update -y

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-dkms=2.0.386.0 --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version, please add '--allow-downgrades' option to
↪'sudo apt-get install'
sudo apt-get install aws-neuron-runtime=1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH
```

(continues on next page)

(continued from previous page)

```
# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum versionlock delete aws-neuron-dkms
sudo yum install aws-neuron-dkms-2.0.386.0 -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################

# Install Neuron Runtime server
# If you are downgrading from newer version , please remove existing package using 'sudo␣
↪yum remove' before installing the older package
sudo yum install aws-neuron-runtime-1.6.9.0 -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
pip install mxnet_neuron==1.5.1.1.6.1.0
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

Tutorials

Computer Vision Tutorials

- ResNet-50 tutorial *[html]* [notebook]
- Model Serving tutorial *[html]*
- Getting started with Gluon tutorial *[html]* [notebook]

10.4 Running Neuron Apache MXNet (Incubating) ResNet50 on Inferentia

10.4.1 Introduction:

In this tutorial we will compile and deploy ResNet50 model for Inferentia. In this tutorial we provide two main sections:

1.Compile the ResNet50 model.

2.Infer the compiled model.

Before running the following verify this Jupyter notebook is running "conda_aws_neuron_mxnet_p36" kernel. You can select the Kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page. Neuron supports Python module, Symbol APIs and the C predict API. The following quick start example uses the Symbol API.

Warning

This tutorial was tested on MXNet-1.5

MXNet-1.5 entered maintenance mode and require Neuron runtime 1.0, please see : MXNet-1.5 enters maintainence mode

To setup development environment for MXNet-1.5 see installation instructions for Neuron 1.15.1 : Neuron-1.15.1 MXNet install

10.4.2 Compile model on Neuron

The following step will compile the resnet50 model. Compilation will take a few minutes on inf1.6xlarge. At the end of compilation, the files resnet-50_compiled-0000.params and resnet-50_compiled-symbol.json will be created in local directory.

[ ]:
```
import mxnet as mx
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
sym, args, aux = mx.model.load_checkpoint('resnet-50', 0)

# Compile for Inferentia using Neuron
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32') }
sym, args, aux = mx.contrib.neuron.compile(sym, args, aux, inputs)
```

(continues on next page)

(continued from previous page)

```
#save compiled model
mx.model.save_checkpoint("resnet-50_compiled", 0, sym, args, aux)
```

[ ]:
```
!ls
```

10.4.3 Deploy on Inferentia

Using same instance to deploy the model.

[ ]:
```
import mxnet as mx
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-
↪server/master/docs/images/kitten_small.jpg?raw=true')
img = mx.image.imread(fname)# convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224) # resize
img = img.transpose((2, 0, 1)) # Channel first
img = img.expand_dims(axis=0) # batchify
img = img.astype(dtype='float32')

sym, args, aux = mx.model.load_checkpoint('resnet-50_compiled', 0)
softmax = mx.nd.random_normal(shape=(1,))
args['softmax_label'] = softmax
args['data'] = img

# Inferentia context
ctx = mx.neuron()

exe = sym.bind(ctx=ctx, args=args, aux_states=aux, grad_req='null')

with open('synset.txt', 'r') as f:
     labels = [l.rstrip() for l in f]

exe.forward(data=img)
prob = exe.outputs[0].asnumpy()# print the top-5
prob = np.squeeze(prob)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
     print('probability=%f, class=%s' %(prob[i], labels[i]))

# Sample output will look like below:
#probability=0.634792, class=n02123045 tabby, tabby cat
#probability=0.193601, class=n02123159 tiger cat
#probability=0.103627, class=n02124075 Egyptian cat
#probability=0.031604, class=n02127052 lynx, catamount
#probability=0.015892, class=n02129604 tiger, Panthera tigris
```

This document is relevant for: `Inf1`

10.5 Tutorial: Neuron Apache MXNet (Incubating) Model Serving

This MXNet Neuron Model Serving (MMS) example is adapted from the MXNet vision service example which uses pretrained squeezenet to perform image classification: https://github.com/awslabs/multi-model-server/tree/master/examples/mxnet_vision.

Before starting this example, please ensure that Neuron-optimized MXNet version mxnet-neuron is installed along with Neuron Compiler.

10.5.1 Warning

If you are using MXNet-1.5, please note that MXNet-1.5 entered maintenance mode and require Neuron Runtime 1.x, please see maintenance_mxnet_1_5. To setup development environment for MXNet-1.5 see installation instructions at mxnet-setup.

If using DLAMI, you can activate the environment aws_neuron_mxnet_p36 and skip the installation part in the first step below.

1. First, install Java runtime and multi-model-server:

```
cd ~/
# sudo yum -y install -q jre # for AML2
sudo apt-get install -y -q default-jre  # for Ubuntu
pip install multi-model-server
```

Download the example code:

```
git clone https://github.com/awslabs/multi-model-server
cd ~/multi-model-server/examples/mxnet_vision
```

2. Compile ResNet50 model to Inferentia target by saving the following Python script to compile_resnet50.py and run "`python compile_resnet50.py`"

```
from packaging import version
import numpy as np
import mxnet as mx

mxnet_version = version.parse(mx.__version__)
if mxnet_version >= version.parse("1.8"):
   import mx_neuron as neuron
else:
   from mxnet.contrib import neuron

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
mx.test_utils.download(path+'synset.txt')

nn_name = "resnet-50"

#Load a model
sym, args, auxs = mx.model.load_checkpoint(nn_name, 0)

#Define compilation parameters
```

(continues on next page)

(continued from previous page)

```
#  - input shape and dtype
inputs = {'data' : mx.nd.zeros([1,3,224,224], dtype='float32') }

# compile graph to inferentia target
csym, cargs, cauxs = neuron.compile(sym, args, auxs, inputs)

# save compiled model
mx.model.save_checkpoint(nn_name + "_compiled", 0, csym, cargs, cauxs)
```

3. Prepare signature file `signature.json` to configure the input name and shape:

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

4. Prepare `synset.txt` which is a list of names for ImageNet prediction classes:

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/squeezenet_v1.1/
↪synset.txt
```

5. Create custom service class following template in model_server_template folder:

```
cp -r ../model_service_template/* .
```

Edit `mxnet_model_service.py` to use the appropriate context.

Make the following change:

```
from packaging import version

mxnet_version = version.parse(mx.__version__)
if mxnet_version >= version.parse("1.8"):
   import mx_neuron as neuron
self.mxnet_ctx = mx.neuron()
```

Comment out the existing context set:

```
#self.mxnet_ctx = mx.cpu() if gpu_id is None else mx.gpu(gpu_id)
```

Also, comment out unnecessary data copy for model_input in `mxnet_model_service.py`:

```
#model_input = [item.as_in_context(self.mxnet_ctx) for item in model_input]
```

6. Package the model with model-archiver:

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
↪handler mxnet_vision_service:handle
```

7. Start MXNet Model Server (MMS) and load model using RESTful API. Please ensure that Neuron RTD is running with default settings (see rtd-getting-started):

```
cd ~/multi-model-server/
multi-model-server --start --model-store examples
# Pipe to log file if you want to keep a log of MMS
curl -v -X POST "http://localhost:8081/models?initial_workers=1&max_workers=1&
↪synchronous=true&url=resnet-50_compiled.mar"
sleep 10 # allow sufficient time to load model
```

Each worker requires a NeuronCore group that can accommodate the compiled model. Additional workers can be added by increasing max_workers configuration as long as there are enough NeuronCores available. Use `neuron-top` to see which models are loaded on specific NeuronCores.

8. Test inference using an example image:

```
curl -O https://raw.githubusercontent.com/awslabs/multi-model-server/master/docs/images/
↪kitten_small.jpg
curl -X POST http://127.0.0.1:8080/predictions/resnet-50_compiled -T kitten_small.jpg
```

You will see the following output:

```
[
  {
    "probability": 0.6375716328620911,
    "class": "n02123045 tabby, tabby cat"
  },
  {
    "probability": 0.1692783385515213,
    "class": "n02123159 tiger cat"
  },
  {
    "probability": 0.12187337130308151,
    "class": "n02124075 Egyptian cat"
  },
  {
    "probability": 0.028840631246566772,
    "class": "n02127052 lynx, catamount"
  },
  {
    "probability": 0.019691042602062225,
    "class": "n02129604 tiger, Panthera tigris"
  }
]
```

9. To cleanup after test, issue a delete command via RESTful API and stop the model server:

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop
```

This document is relevant for: `Inf1`

10.6 MXNet 1.8: Getting Started with Gluon Tutorial

In this tutorial you will compile and deploy resnet-50 using the newly supported MXNet 1.8 and Gluon API on an Inf1 instance. This tutorial is only supported with MXNet 1.8.

This Jupyter notebook should be run on an inf1.6xlarge instance since you will be loading and compiling several large models.

To run this tutorial, please make sure you deactivate any existing MXNet conda environments you already using. Install MXNet 1.8 by following the instructions at MXNet Setup Guide. You would also need to change your kernel to use the correct Python environment setup earlier by clicking Kerenel->Change Kernel->Python (Neuron MXNet)

10.7 Compile

A trained model must be compiled to Inferentia target before it can run on Inferentia. In this step we compile a pre-trained ResNet50 and export it as a compiled MXNet checkpoint.

Compilation will take a few minutes. At the end of compilation, the files resnet-50_compiled-0000.params and resnet-50_compiled-symbol.json will be created in local directory.

To check the supported operations for the uncompiled model or information on Neuron subgraphs for the compiled model, please see Neuron Check Model.

[ ]:
```
import os
import mxnet as mx
import mx_neuron as neuron
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
block = mx.gluon.nn.SymbolBlock.imports('resnet-50-symbol.json',\
    ['data', 'softmax_label'], 'resnet-50-0000.params', ctx=mx.cpu())

block.hybridize()

# Compile for Inferentia using Neuron
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32'), 'softmax_
↪label' : mx.nd.ones([1], name='data', dtype='float32') }
block = neuron.compile(block, inputs=inputs)

#save compiled model
block.export("resnet-50_compiled", 0, block)
```

[ ]:
```
!ls
```

10.8 Deploy

Deply on Infenrentia to see the inference results as below:

```
probability=0.643591, class=n02123045 tabby, tabby cat
probability=0.184392, class=n02123159 tiger cat
probability=0.105063, class=n02124075 Egyptian cat
probability=0.030101, class=n02127052 lynx, catamount
probability=0.016112, class=n02129604 tiger, Panthera tigris
```

[ ]:
```
import numpy as np
import mxnet as mx
import mx_neuron as neuron

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-
↪server/master/docs/images/kitten_small.jpg?raw=true')
img = mx.image.imread(fname)# convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224) # resize
img = img.transpose((2, 0, 1)) # Channel first
img = img.expand_dims(axis=0) # batchify
img = img.astype(dtype='float32')

block = mx.gluon.nn.SymbolBlock.imports('resnet-50_compiled-symbol.json',\
    ['data', 'softmax_label'], 'resnet-50_compiled-0000.params', ctx=mx.cpu())
softmax = mx.nd.random_normal(shape=(1,))

out = block(img, softmax).asnumpy()

with open('synset.txt', 'r') as f:
    labels = [l.rstrip() for l in f]

out = block(img, softmax).asnumpy()

prob = np.squeeze(out)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
    print('probability=%f, class=%s' %(prob[i], labels[i]))
```

Natural Language Processing (NLP) Tutorials

- MXNet 1.8: Using data parallel mode tutorial *[html]* [notebook]

10.9 Using Data Parallel Mode with Gluon MXNet

In this tutorial, you will compile a Gluon BERT model and run in data-parallel mode to completely utilize the NeuronCores. Here you will benchmark a multi-worker setup and compare it with a single worker.

This tutorial is intended only for MXNet-1.8.

In this tutorial, we will be using an inf1.2xlarge with the latest AWS Deep Learning AMI (DLAMI). The inf1.2xlarge instance has 1 AWS Inferentia Chip with 4 NeuronCores.

10.9.1 Setting up your environment

To run this tutorial, please make sure you deactivate any existing MXNet conda environments you already using. Install MXNet 1.8 by following the instructions at MXNet Setup Guide. You would also need to change your kernel to use the correct Python environment setup earlier by clicking Kerenel->Change Kernel->Python (Neuron MXNet)

10.9.2 Install dependencies

We have to install gluon-nlp to get the BERT model. Run the following command to install:

[ ]:
```
!python -m pip install gluonnlp
```

10.9.3 Compiling BERT Model

Next, we compile the Gluon BERT model and save it. Once the model is compiled, we use the same model across the entire tutorial. In this tutorial, we will be using a BERT model with sequence length 32

[ ]:
```
import os
import mxnet as mx
import mx_neuron
import gluonnlp as nlp
```

[ ]:
```
BERT_MODEL = 'bert_12_768_12'
BERT_DATA = 'book_corpus_wiki_en_uncased'
batch_size = 1
seq_len = 32
num_cores = 1
dtype = 'float32'

compiled_model_path = '{}.compiled.{}.{}'.format(BERT_MODEL, batch_size, seq_len)

model, vocab = nlp.model.get_model(BERT_MODEL,
                                   dataset_name=BERT_DATA,
                                   use_classifier=False,
                                   use_decoder=False, ctx=mx.cpu())
```

(continues on next page)

(continued from previous page)

```
# Create sample inputs for compilation
words = mx.nd.ones([batch_size, seq_len], name='words', dtype=dtype)
valid_len = mx.nd.ones([batch_size,], name='valid_len', dtype=dtype)
segments = mx.nd.ones([batch_size, seq_len], name='segments', dtype=dtype)
inputs = {'data0': words, 'data1': segments, 'data2': valid_len}

# Compiler Args ~~
options = {}
embeddingNames = ['bertmodel0_word_embed_embedding0_fwd', 'bertmodel0_token_type_embed_
↪embedding0_fwd', 'bertencoder0_embedding0']
options.update({'force_incl_node_names': embeddingNames})
options.update({'flags': ['--fp32-cast matmult']})

# Compile and save ~~
model = mx_neuron.compile(model, inputs=inputs, **options)
model.export(compiled_model_path)
```

10.9.4 Data Parallel Mode

Data Parallel Mode is a setup in which you launch multiple copies of the same model, such that each model is running independently of the other. In other words, each model has its own resources to run inference.

On an inf1.2xlarge instance, we have 4 NeuronCores. Hence, we can launch 4 models such that each model is loaded on a single NeuronCore. This unables us to process 4 request concurrently without linear increase in latency. As a result, the throughput of the system increases when compared to a single model inference. This would also allow us to utilize all the 4 NeuronCores on the instance.

Run through the next set of cells to see the difference in throughput as we scale from one model to 4 models running in parallel.

[ ]:
```
import numpy as np

def get_sample_inputs(batch_size, seq_len):
    words = np.ones([batch_size, seq_len], dtype=np.float32)
    valid_len = np.ones([batch_size,], dtype=np.float32)
    segments = np.ones([batch_size, seq_len], dtype=np.float32)
    inputs = {'data0': words, 'data1': segments, 'data2': valid_len}
    return inputs
```

Next for comparison purposes, we run the setup with 1 worker. To do this, we set the num_cores=1. This would launch only 1 model running on a single NeuronCore. After running the below cell, note down the latency and throughput for the system

[ ]:
```
from parallel import NeuronSimpleDataParallel
from benchmark_utils import Results
import time
import functools
import os
import numpy as np
import warnings
```

(continues on next page)

(continued from previous page)

```
num_cores = 1
batch_size=1

# Each worker process should use one core, hence we set
#    os.environ['NEURON_RT_NUM_CORES'] = "1"
os.environ["NEURON_RT_NUM_CORES"] = "1"

#Result aggregation class (code in bert_benchmark_utils.py)
results = Results(batch_size, num_cores)
def result_handler(output, start, end):
    elapsed = end - start
    results.add_result([elapsed], [end], [start])

inputs = get_sample_inputs(batch_size, seq_len)
parallel_neuron_model = NeuronSimpleDataParallel(compiled_model_path, num_cores, inputs)

#Starting the inference threads
parallel_neuron_model.start_continuous_inference()

# Warm up the cores
for _ in range(num_cores*4):
    parallel_neuron_model.warmup(inputs)

# Need to run for high number of iterations to benchmark the models
for _ in range(1000):
    parallel_neuron_model.infer(inputs)
    # Passing the result_handler as a callback function
    parallel_neuron_model.add_result(result_handler)

# Stop inference
parallel_neuron_model.stop()
# Since we are using a multi-process execution with a shared queue, some inferences
# may still be in execution phase. Hence we need to wait till all the inputs are␣
↪processed
# add_all_results() will collect all the results of requests which are in this state
parallel_neuron_model.add_all_results(result_handler)


with open("benchmark.txt", "w") as f:
    results.report(f, window_size=1)

with open("benchmark.txt", "r") as f:
    for line in f:
        print(line)
```

Now we run the setup with 4 workers. To do this, we set the num_cores=4. This would launch 4 model running each running on individual NeuronCore. All the 4 models are running in individual processes, in other words the models are running in parallel.

To feed the models efficiently, we use the producer-consumer setup, in which all processes running a model act as consumers. All consumers are fed using a sharing input queue.

Now we run the below setup. You may notice, that the throughput increase by >2x when compared to a single worker setup.

[ ]:
```
from parallel import NeuronSimpleDataParallel
from benchmark_utils import Results
import time
import functools
import os
import numpy as np

num_cores = 4
batch_size=1

os.environ["NEURON_RT_NUM_CORES"] = "1"

#Result aggregation class (code in bert_benchmark_utils.py)
results = Results(batch_size, num_cores)
def result_handler(output, start, end):
    elapsed = end - start
    results.add_result([elapsed], [end], [start])

inputs = get_sample_inputs(batch_size, seq_len)
parallel_neuron_model = NeuronSimpleDataParallel(compiled_model_path, num_cores, inputs)

#Starting the inference threads
parallel_neuron_model.start_continuous_inference()

# Warm up the cores
for _ in range(num_cores*4):
    parallel_neuron_model.warmup(inputs)

# Need to run for high number of iterations to benchmark the models
for _ in range(5000):
    parallel_neuron_model.infer(inputs)
    # Passing the result_handler as a callback function
    parallel_neuron_model.add_result(result_handler)

# Stop inference
parallel_neuron_model.stop()
# Since we are using a multi-process execution with a shared queue, some inferences
# may still be in execution phase. Hence we need to wait till all the inputs are␣
↪processed
# add_all_results() will collect all the results of requests which are in this state
parallel_neuron_model.add_all_results(result_handler)


with open("benchmark.txt", "w") as f:
    results.report(f, window_size=1)

with open("benchmark.txt", "r") as f:
    for line in f:
        print(line)
```

Utilizing Neuron Capabilities Tutorials

- NeuronCore Groups tutorial *[html]* [notebook]

10.10 Neuron Apache MXNet (Incubating) - Configurations for NeuronCore Groups Using Resnet50

10.10.1 Introduction:

In this tutorial we will compile and deploy Resnet-50 model in parallel using the concept of NeuronCore Groups on an Inf1 instance. This Jupyter notebook should be run on an instance which is inf1.6xlarge or larger. For simplicity we will run this tutorial on inf1.6xlarge but in real life scenario the compilation should be done on a compute instance and the deployment on inf1 instance to save costs.

Set environment variable NEURON_RT_NUM_CORES to the total number of Neuron cores that will be utilized. The consecutive NeuronCore groups will be created by Neuron Runtime and place the models to the cores according to the compiled size.

Note that in order to map a model to a group, the model must be compiled to fit within the group size. To limit the number of NeuronCores during compilation, use compiler_args dictionary with field "–neuroncore-pipeline-cores" set to the group size. For exmaple, if NEURON_RT_NUM_CORES=4 and two models compiled with "–neuroncore-pipeline-cores=3" and "–neuroncore-pipeline-cores=1" were loaded, the first model would occupy NC0-2 and the second model would occupy NC3.

```
compile_args = {'--neuroncore-pipeline-cores' : 2}
sym, args, auxs = neuron.compile(sym, args, auxs, inputs, **compile_args)
```

In this tutorial we provide two main sections:

1. Compile the Resnet50 model for Neuron
2. Run inference using NeuronCore Groups

Please use environment `conda_aws_neuron_mxnet_p36`.

10.10.2 Compile model for Neuron

Model must be compiled to Inferentia target before it can be used on Inferentia. In the following we will compile the the flag, –neuroncore-pipeline-cores set to 2 and run it. The files resnet-50_compiled-0000.params and resnet-50_compiled-symbol.json will be created in local directory

[ ]:
```
from packaging import version
import mxnet as mx
import numpy as np

import mx_neuron as neuron

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
sym, args, aux = mx.model.load_checkpoint('resnet-50', 0)

# Compile for Inferentia using Neuron, fit to NeuronCore group size of 2
```

(continues on next page)

(continued from previous page)

```
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32') }
compile_args = {'--neuroncore-pipeline-cores' : 2}
sym, args, aux = neuron.compile(sym, args, aux, inputs, **compile_args)

#save compiled model
mx.model.save_checkpoint("resnet-50_compiled", 0, sym, args, aux)
```

10.10.3 Run inference using NeuronCore Groups

Within the framework, the model can be mapped to specific cores using `ctx=mx.neuron(N)` context where N specifies the index of the Neuron core to deploy. For more information, see https://awsdocs-neuron.readthedocs-hosted.com/en/latest/neuron-guide/appnotes/perf/flex-eg.html .

[ ]:
```
import os
import warnings

mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-
↪server/master/docs/images/kitten_small.jpg?raw=true')
img = mx.image.imread(fname) # convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224) # resize
img = img.transpose((2, 0, 1)) # Channel first
img = img.expand_dims(axis=0) # batchify
img = img.astype(dtype='float32')

sym, args, aux = mx.model.load_checkpoint('resnet-50_compiled', 0)
softmax = mx.nd.random_normal(shape=(1,))
args['softmax_label'] = softmax
args['data'] = img

os.environ["NEURON_RT_NUM_CORES"] = '4'


# Inferentia context - group index 1 (size 2) would skip NC0 and place the
# compiled model onto NC1,2
ctx = mx.neuron(1)

exe = sym.bind(ctx=ctx, args=args, aux_states=aux, grad_req='null')

with open('synset.txt', 'r') as f:
     labels = [l.rstrip() for l in f]

exe.forward(data=img)
prob = exe.outputs[0].asnumpy()# print the top-5
prob = np.squeeze(prob)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
     print('probability=%f, class=%s' %(prob[i], labels[i]))
```

You can experiment with different Neuron core group combinations and different models.

Troubleshooting

If not enough NeuronCores are provided, an error message will be displayed:

```
mxnet.base.MXNetError: [04:01:39] src/operator/subgraph/neuron/./neuron_util.h:541:␣
↪Check failed: rsp.status().code() == 0: Failed load model with Neuron-RTD Error.␣
↪Neuron-RTD Status Code: 9, details: ""
```

Note: To use Jupyter Notebook see:

- setup-jupyter-notebook-steps-troubleshooting
- running-jupyter-notebook-as-script

API Reference Guide

This document is relevant for: `Inf1`

10.11 Neuron Apache MXNet (Incubating) Compilation Python API

The MXNet-Neuron compilation Python API provides a method to compile model graph for execution on Inferentia.

10.11.1 Description

Within the graph or subgraph, the compile method selects and sends Neuron-supported operations to Neuron-Compiler for compilation and saves the compiled artifacts in the graph. Uncompilable operations are kept as original operations for framework execution.

The compiled graph can be saved using the MXNet save_checkpoint and served using MXNet Model Serving. Please see *Tutorial: Neuron Apache MXNet (Incubating) Model Serving* for more information about exporting to saved model and serving using MXNet Model Serving.

Options can be passed to Neuron compiler via the compile function. For example, the "`--neuroncore-pipeline-cores`" option directs Neuron compiler to compile each subgraph to fit in the specified number of NeuronCores. This number can be less than the total available NeuronCores on an Inf1 instance. See *Neuron compiler CLI Reference Guide (neuron-cc)* for more information about compiler options.

For debugging compilation, use SUBGRAPH_INFO=1 environment setting before calling the compilation script. The extract subgraphs are preserved as hidden files in the run directory. For more information, see *Using Neuron GatherInfo Tool to collect debug and support information*

10.11.2 MXNet 1.5

10.11.3 Method

```
from mxnet.contrib import neuron
neuron.compile(sym, args, aux, inputs, **compile_args)
```

10.11.4 Arguments

- **sym** - Symbol object loaded from symbol.json file
- **args** - args/params dictionary loaded from params file
- **aux** - aux/params dictionary loaded from params file
- **inputs** - a dictionary with key/value mappings for input name to input numpy arrays
- **kwargs** (optional) - a dictionary with key/value mappings for MXNet-Neuron compilation and Neuron Compiler options.
  - For example, to limit the number of NeuronCores per subgraph, use `compile_args={'--neuroncore-pipeline-cores' : N}` where N is an integer representing the maximum number of NeuronCores per subgraph.
  - Additional compiler flags can be passed using `'flags' : [<flags>]` where is a comma separated list of strings. See *Using Neuron GatherInfo Tool to collect debug and support information* for example of passing debug flags to compiler.
  - Advanced option to exclude node names: `compile_args={'excl_node_names' : [<node names>]}` where is a comma separated list of node name strings.

10.11.5 Returns

- **sym** - new partitioned symbol
- **args** - modified args/params
- **auxs** - modified aux/params

10.11.6 Example Usage: Compilation

The following is an example usage of the compilation, with default compilation arguments:

```
from mxnet.contrib import neuron
...
neuron.compile(sym, args, aux, inputs={'data' : img})
```

10.11.7 MXNet 1.8

10.11.8 Method

```
import mx_neuron as neuron
neuron.compile(obj, args=None, aux=None, inputs=None, **compile_args)
```

10.11.9 Arguments

- **obj** - Symbol object loaded from symbol.json file or gluon.HybridBlock object
- **args** (optional) - args/params dictionary loaded from params file. Only needed in case of Symbol object
- **aux** (optional) - aux/params dictionary loaded from params file. Only needed in case of Symbol object
- **inputs** - a dictionary with key/value mappings for input name to input numpy arrays.
- **kwargs** (optional) - a dictionary with key/value mappings for MXNet-Neuron compilation and Neuron Compiler options.
  - For example, to limit the number of NeuronCores per subgraph, use `compile_args={'--neuroncore-pipeline-cores' : N}` where N is an integer representing the maximum number of NeuronCores per subgraph.
  - Additional compiler flags can be passed using `'flags' : [<flags>]` where is a comma separated list of strings. See *Using Neuron GatherInfo Tool to collect debug and support information* for example of passing debug flags to compiler.
  - Advanced option to exclude node names: `compile_args={'excl_node_names' : [<node names>]}` where is a comma separated list of node name strings.
  - work_dir: relative or absolute path for storing compiler artifacts (including params and jsons) generated during compilation when SUBGRAPH_INFO=1.

10.11.10 Returns

- **(sym, args, auxs)** - for symbol object as input. sym, args and auxs are new partitioned symbol, modified args/params and modified aux/params repectively.
- **(obj)** - for gluon.HybridBlock object as input. obj is the parititioned and optimized gluon.Hybrid block object for Neuron backend.

10.11.11 Example Usage: Compilation

The following is an example usage of the compilation, with default compilation arguments for symbol object:

```
import mx_neuron as neuron
...
neuron.compile(sym, args, aux, inputs={'data' : img})
```

The following is an example usage of the compilation, with default compilation arguments for gluon.HybridBlock object (only supported in MXNet-Neuron 1.8):

```
import mx_neuron as neuron
...
neuron.compile(obj, inputs={'data' : img})
```

10.11.12 Example Usage: Extract Compilation Statistics

To extract operation counts, insert the following code after compile step (assume csym is the compiled MXNet symbol):

```
import json

# Return list of nodes from MXNet symbol
def sym_nodes(sym):
  return json.loads(sym.tojson())['nodes']

# Return number of operations in node list
def count_ops(graph_nodes):
  return len([x['op'] for x in graph_nodes if x['op'] != 'null'])

# Return triplet of compile statistics
# - count of operations in symbol database
# - number of Neuron subgraphs
# - number of operations compiled to Neuron runtime
def get_compile_stats(sym):
  cnt = count_ops(sym_nodes(sym))
  neuron_subgraph_cnt = 0
  neuron_compiled_cnt = 0
  for g in sym_nodes(sym):
    if g['op'] == '_neuron_subgraph_op':
      neuron_subgraph_cnt += 1
      for sg in g['subgraphs']:
        neuron_compiled_cnt += count_ops(sg['nodes'])
  return (cnt, neuron_subgraph_cnt, neuron_compiled_cnt)

original_cnt = count_ops(sym_nodes(sym))
post_compile_cnt, neuron_subgraph_cnt, neuron_compiled_cnt = get_compile_stats(csym)
print("INFO:mxnet: Number of operations in original model: ", original_cnt)
print("INFO:mxnet: Number of operations in compiled model: ", post_compile_cnt)
print("INFO:mxnet: Number of Neuron subgraphs in compiled model: ", neuron_subgraph_cnt)
print("INFO:mxnet: Number of operations placed on Neuron runtime: ", neuron_compiled_cnt)
```

```
INFO:mxnet: Number of operations in original model:  67
INFO:mxnet: Number of operations in compiled model:  4
INFO:mxnet: Number of Neuron subgraphs in compiled model:  2
INFO:mxnet: Number of operations placed on Neuron runtime:  65
```

This document is relevant for: `Inf1`

Developer Guide

This document is relevant for: `Inf1`, `Trn1`

10.12 Flexible Execution Group (FlexEG) in Neuron-MXNet

10.12.1 Introduction

Inf1 instances are available with a different number of Inferentia chips, each Inferentia chip is combined of 4 NeuronCores and an Inf1 instance includes 4 to 64 NeuronCores depending on the instance size. With Neuron Runtime 1.x (neuron-rtd server), NeuronCores could be combined into NeuronCore Groups (NCG), which were basic scheduling units of compiled neural network in Neuron. Creation of desired sized NCGs was done at the start of the application and could not be modified afterwards.

Starting with Neuron SDK 1.16.0, and with the introduction of Neuron Runtime 2.x, MXNet Neuron 1.8 introduces Flexible Execution Groups (FlexEG) feature. With FlexEG, you do not have to create NCGs at the start of the process, instead you will set the index of the first NeuronCore you want to load models onto, and FlexEG feature will enable the flexibility of loading models onto any available NeuronCore on the inf1 instance starting from the first NeuronCore you set. This guide will show you how to efficiently utilize NeuronCores using FlexEG feature in NeuronMXNet.

10.12.2 FlexEG

With the introduction of FlexEG, you don't need to create NCGs and can load models onto a group of consecutive NeuronCores by providing the index of the first NeuronCore in the group. Neuron runtime takes care of figuring out the number of NeuronCores required for the given compiled model and loads the model using the required number of cores (sequentially starting with the NeuronCore index provided by the user).

For example, assuming that you have an Inf1.6xl machine and there are 4 models A, B, C, D compiled to 2, 4, 3, and 4 NeuronCores respectively, you can map any model to any core by context `mx.neuron(neuron_core_index)` where `neuron_core_index` is the NeuronCore index (0,1,2,3,4 …).

In the example below, you map model A to `mx.neuron(0)` context, model B to `mx.neuron(2)` context, model C to `mx.neuron(6)` context and model D to `mx.neuron(9)` context.

The above configuration is achieved by using application code similar to below:

```
# Load models (MXNet)
# loaded into the 2 cores starting with core 0
sym, args, aux = mx.model.load_checkpoint(mx_model0_file, 0)
model0 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')
# loaded into the 4 cores starting with core 2
sym, args, aux = mx.model.load_checkpoint(mx_model1_file, 0)
model1 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')
# loaded into the 3 cores starting with core 6
sym, args, aux = mx.model.load_checkpoint(mx_model2_file, 0)
model2 = sym.bind(ctx=mx.neuron(6), args=args, aux_states=aux, grad_req='null')
# loaded into the 4 cores starting with core 9
sym, args, aux = mx.model.load_checkpoint(mx_model3_file, 0)
model3 = sym.bind(ctx=mx.neuron(9), args=args, aux_states=aux, grad_req='null')

# run inference by simply calling the loaded model
results0 = model0.forward(data=inputs0)
results1 = model1.forward(data=inputs1)
results2 = model2.forward(data=inputs2)
results3 = model3.forward(data=inputs3)
```

Since there is no NCG creation at the start of the process, you can load the same four models but in a different configuration by changing the context being used for inference. For example, you could map model C to `mx.neuron(0)` context, model A to `mx.neuron(3)` context, model D to `mx.neuron(5)` context and model B to `mx.neuron(9)` context.

10.12.3 Migration from NeuronCore Groups to FlexEG

NeuronCore Groups are defined by setting the environment variable `NEURONCORE_GROUP_SIZES` with a comma separated list of number of cores in each group. In this mode of operation, number of devices (defined in `NEURONCORE_GROUP_SIZES`) are grouped together to create a single entity.

`NEURONCORE_GROUP_SIZES` environment variable is set at runtime:

```
#!/bin/bash
export NEURONCORE_GROUP_SIZES=2,4,3,4
python your_neuron_application.py
```

NeuronCore groups are created once at the start of the application and cannot be modified / re-created till the application process runs. The above flow creates 4 neuron devices with 2,4,3 and 4 devices each. In order to get the same

configuration as the example from before , you map model A to `mx.neuron(0)` context, model B to `mx.neuron(1)` context, model C to `mx.neuron(2)` context and model D to `mx.neuron(3)` context.

This can be achieved programmatically as shown below:

```
# Set Environment
os.environ['NEURONCORE_GROUP_SIZES']='2,4,3,4'

# Load models (MXNet)
# loaded into the first group of NC0-NC1
sym, args, aux = mx.model.load_checkpoint(mx_model0_file, 0)
model0 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')
# loaded into the second group of NC2-NC5
sym, args, aux = mx.model.load_checkpoint(mx_model1_file, 0)
model1 = sym.bind(ctx=mx.neuron(1), args=args, aux_states=aux, grad_req='null')
# loaded into the third group of NC6-NC8
sym, args, aux = mx.model.load_checkpoint(mx_model2_file, 0)
model2 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')
# loaded into the fourth group of NC9-NC12
sym, args, aux = mx.model.load_checkpoint(mx_model3_file, 0)
model3 = sym.bind(ctx=mx.neuron(3), args=args, aux_states=aux, grad_req='null')

# run inference by simply calling the loaded model
results0 = model0.forward(data=inputs0)
results1 = model1.forward(data=inputs1)
results2 = model2.forward(data=inputs2)
results3 = model3.forward(data=inputs3)
```

So comparing to FlexEG, we see that in case of NCGs neuron context requires the index of the execution group, while in FlexEG neuron context requires the NeuronCore index of the first NeuronCore on which the model is supposed to be loaded and executed. For example, with `NEURONCORE_GROUP_SIZES='2,4,3,4'`, `ctx=mx.neuron(1)` loads the model on execution group 1 which effectively loads the model on the 2nd NCG group which has 4 NeuronCores.

10.12.4 Best practices when using FlexEG

FlexEG gives the user most flexibility in terms of accessing cores and loading models on specific cores. With this the users can effortlessly load and execute new models on NeuronCores without closing the application. Here we shall outline some of the best practices that should be kept in mind while using FlexEG.

Choosing starting core

FlexEG tries to use the required number of cores (based on the input model) starting with the core index provided by the user. Incase the system, doesnt have the required number of cores after the starting core index, model load will fail. For example: We have a model X which needs 2 cores and an inf1.xl machine with 4 NeuronCores (NeuronCore indexes are: 0, 1, 2 and 3). As the model needs at least 2 cores, valid start indexes for this model are: 0, 1, 2. However if the user gives 3 as the neuron context, then there are no 2 cores available starting from core 3. So it will fail.

Performance vs. Flexibility tradeoff

While using data parallel model of operation (were models are executed in parallel), for optimal performance the user should make sure that the models are not sharing any cores. That is because NeuronCores can execute one model at a time, when two or more models are executed on the same core (assuming that they are already loaded), it executes the first model, stops it, starts the second model and then executes it. This is called model switching and involves additional overhead and prevents execution on model in parallel. For example: assuming that you have an Inf1.6xl machine and there are 4 models A, B, C, D compiled to 2, 4, 3, and 4 NeuronCores respectively. Loading model A to `mx.neuron(0)` context, model B to `mx.neuron(2)` context, model C to `mx.neuron(6)` context and model D to `mx.neuron(9)` context is a good configuration because no two models are sharing NeuronCores and thus can be executed in parallel. However, Loading model A to `mx.neuron(0)` context, model B to `mx.neuron(2)` context, model C to `mx.neuron(5)` context and model D to `mx.neuron(9)` context is a not a good configuration as models B and C share NeuronCore 5 and thus cannot be executed in parallel.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`

10.13 Troubleshooting Guide for Neuron Apache MXNet (Incubating)

Table of Contents

- *Inference Runtime Error*
  - *Out-of-memory error when calling Symbol API bind() too many times*
  - *Inference crashed with MXNetError: InferShapeKeyword argument name xyz not found*
  - *Inference crashed at mx.nd.waitall() with MXNetError: Check failed: bin.dtype() == mshadow::kUint8*
  - *Inference crashed with NRTD error 1002*
- *Multi-Model Server*
  - *Failed to create NEURONCORE Group with GRPC Error. Status Error: 14, Error message: "Connect Failed"*
  - *Multiple MMS workers die with "Backend worker process die." message*
  - *MMS throws a "mxnet.base.MXNetError: array::at" error*
  - *MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded*
  - *Pipeline mode is not able to execute inferences requests in parallel*
  - *Features only in MXNet-Neuron 1.5*
  - *Features only in MXNet-Neuron 1.8*

10.13.1 Inference Runtime Error

Out-of-memory error when calling Symbol API bind() too many times

Important: `NEURONCORE_GROUP_SIZES` will no longer be supported starting Neuron 1.19.0 release if your application is using `NEURONCORE_GROUP_SIZES` please see *Migrate your application to Neuron Runtime 2.x (libnrt.so)* and eol-ncgs-env_2 for more details.

If you see out-of-memory error when using Symbol API's bind() function, please ensure that the bind() function is called once for each desired model instance. For example, on inf1.xlarge, use Symbol API to create 4 parallel instances of a model that was compiled to 1 NeuronCore (–neuroncore-pipeline-cores=1), each is bound to an different mx.neuron(i) context where i is the NeuronCore Group index ranging from 0 to 3. Then use 4 threads to feed the 4 instances in parallel. For example:

```
NUM_PARALLEL = 4
os.environ['NEURONCORE_GROUP_SIZES'] = ','.join('1' for _ in range(NUM_PARALLEL))

data_iter = []
for i in range(NUM_PARALLEL):
    data_iter.append(mx.io.ImageRecordIter(
        path_imgrec=recfile_base, data_shape=(3, 224, 224), batch_size=1,
        prefetch_buffer=1,
        num_parts=NUM_PARALLEL, part_index=i))
```

(continues on next page)

(continued from previous page)

```
sym, args, auxs = mx.model.load_checkpoint('resnet-50_compiled', 0)

exec_list = []
for i in range(NUM_PARALLEL):
    exec = sym.bind(ctx=mx.neuron(i), args=args, aux_states=auxs, grad_req='null')
    exec_list.append(exec)

def single_thread_infer(i):
    for batch in data_iter[i]:
        img = batch.data[0]
        label = batch.label
        feed_dict = {'data': img}
        exe = exec_list[i]
        exe.copy_params_from(feed_dict)
        exe.forward()
        out = exe.outputs[0]

future_list = []
with futures.ThreadPoolExecutor(max_workers=NUM_PARALLEL) as executor:
    for i in range(NUM_PARALLEL):
        future_list.append(executor.submit(single_thread_infer, i))
```

Inference crashed with MXNetError: InferShapeKeyword argument name xyz not found

If you see MXNetError:

```
mxnet.base.MXNetError: [11:55:39] src/c_api/c_api_symbolic.cc:508: InferShapeKeyword␣
↪argument name xyz not found."
```

This is followed by a list of "Candidate arguments". This list shows all the input argument names that the model knows about, and 'xyz' is not in the list. To fix this, remove entry xyz from the feed dictionary.

Inference crashed at mx.nd.waitall() with MXNetError: Check failed: bin.dtype() == mshadow::kUint8

When executing Symbol API's forward function followed by mx.nd.waitall(), where MXNetError exception occurs with 'Check failed: bin.dtype() == mshadow::kUint8'.

Inference crashed with NRTD error 1002

During inference, the user may encounter an error with details "[NRTD:infer_wait] error: 1002":

```
mxnet.base.MXNetError: [11:26:56] src/operator/subgraph/neuron/./neuron_util.h:1175:␣
↪Check failed: rsp_wait.status().code() == 0 || rsp_wait.status().code() == 1003: Failed
Infer Wait with Neuron-RTD Error. Neuron-RTD Status Code: 1002, details: "[NRTD:infer_
↪wait] error: 1002
"
```

Runtime errors are listed in rtd-return-codes. In particular, 1002 means that some invalid input has been submitted to infer, e.g. missing some of the input tensors, incorrect input tensor sizes. Please examine /var/log/syslog to see imore details on the error. For example, you may see:

```
Oct 30 19:13:39 ip-172-31-93-131 nrtd[1125]: [TDRV:io_queue_prepare_input_nonhugetlb]␣
↪Unexpected input size, for data00, expected: 2097152, received: 33554432
```

This means that the input tensor size is larger than what the model was compiled for (i.e. the example input tensor shapes passed during compilation.

10.13.2 Multi-Model Server

Failed to create NEURONCORE Group with GRPC Error. Status Error: 14, Error message: "Connect Failed"

NOTE: This error only applies to MXNet 1.5.

If the client is unable to start workers and you get a message that MMS is unable to create NeuronCore Group, please check that Neuron RTD is running (neuron-rtd process).

```
{
"code": 500,
"type": "InternalServerException",
"message": "Failed to start workers"
}
```

```
2019-10-23 19:56:23,187 [INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.
↪mms.wlm.WorkerLifeCycle - [19:56:23] src/operator/subgraph/inferentia/./inferentia_
↪util.h:218: Check failed: status.ok() Failed to create NeuronCore Group with GRPC␣
↪Error. Status Error: 14, Error message: "Connect Failed"
```

Multiple MMS workers die with "Backend worker process die." message

Important: NEURONCORE_GROUP_SIZES will no longer be supported starting Neuron 1.19.0 release if your application is using NEURONCORE_GROUP_SIZES please see *Migrate your application to Neuron Runtime 2.x (libnrt.so)* and eol-ncgs-env_2 for more details.

If you run inference with MMS and get multiple messages "Backend worker process die", please ensure that the number of workers ("intial_workers") passed during load model is less than or equal to number of NeuronCores available divided by number of NeuronCores required by model.

```
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - Backend worker process die.
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - Traceback (most recent call last):
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - File "/usr/local/lib/python3.6/site-packages/
↪mxnet/symbol/symbol.py", line 1524, in simple_bind
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - ctypes.byref(exe_handle)))
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - File "/usr/local/lib/python3.6/site-packages/
↪mxnet/base.py", line 252, in check_call
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - raise MXNetError(py_str(_LIB.
↪MXGetLastError()))
com.amazonaws.ml.mms.wlm.WorkerLifeCycle - mxnet.base.MXNetError: [00:26:32] src/
↪operator/subgraph/neuron/./neuron_util.h:221: Check failed: 0 == create_eg_rsp.
↪status().code() Failed to create NeuronCore Group with KRTD Error. KRTD Status Code: 4,
↪ details: ""
```

As indicated in *Performance Tuning*, for greater flexibility user can use NEURONCORE_GROUP_SIZES to specify the groupings of NeuronCores into Neuron devices, each device consisting of one or more NeuronCores. Each worker would take a device. The total number of NeuronCores taken by all the workers should be less than or equal the total number of NeuronCores visible to neuron-rtd. This situation should be considered at full load (MMS scales up to max_workers). Additionally, to properly assign model to Neuron device, the environment NEURONCORE_GROUP_SIZES must be specified within the model server class (ie. mxnet_model_service.py in the example above). For example, add the following line within mxnet_model_service.py for model compiled to 1 NeuronCore:

```
os.environ['NEURONCORE_GROUP_SIZES'] = '1'
```

More information about max_worker limit setting can be found at MMS Management API Documentation. For example, to run up to 4 workers in inf1.xlarge where 4 NeuronCores are available by default to Neuron-RTD, set max_workers to 4:

```
curl -v -X PUT "http://localhost:8081/models/squeezenet_v1.1_compiled?min_worker=1?max_
↪worker=4"
```

MMS throws a "mxnet.base.MXNetError: array::at" error

If you see "mxnet.base.MXNetError: array::at" when running MMS please check that NDArray/Gluon API is not used as they are not supported in MXNet-Neuron. If you would like to use NDArray or Gluon API, please upgrade to MXNet 1.8.

```
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪ array::at
[INFO ] W-9000-squeezenet_v1.1_compiled com.amazonaws.ml.mms.wlm.WorkerThread - Backend␣
↪response time: 30
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪ Traceback (most recent call last):
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪   File "/tmp/models/6606fa046f68a34df87f15362a7a2d9a49749878/model_handler.py", line␣
↪82, in handle
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪     data = self.inference(data)
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪   File "/tmp/models/6606fa046f68a34df87f15362a7a2d9a49749878/mxnet_model_service.py",␣
↪line 153, in inference
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪     d.wait_to_read()
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪   File "/home/user/regression_venv_p3.6/lib/python3.6/site-packages/mxnet/ndarray/
↪ndarray.py", line 1819, in wait_to_read
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪     check_call(_LIB.MXNDArrayWaitToRead(self.handle))
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪   File "/home/user/regression_venv_p3.6/lib/python3.6/site-packages/mxnet/base.py",␣
↪line 253, in check_call
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪     raise MXNetError(py_str(_LIB.MXGetLastError()))
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪ mxnet.base.MXNetError: array::at
[INFO ] W-9000-squeezenet_v1.1_compiled-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle -
↪ Invoking custom service failed.
```

MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded

NOTE: This issue is resolved in version 1.5.1.1.1.88.0 released 11/17/2020 and only applies for MXNet 1.5.

MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server. Restarting the model server may fail with "Failed to create NEURONCORE_GROUP" error:

```
mxnet.base.MXNetError: [00:26:59] src/operator/subgraph/neuron/./neuron_util.h:348:␣
↪Check failed:    0 == create_eg_rsp.status().code(): Failed to create NEURONCORE_GROUP␣
↪with Neuron-RTD Error. Neuron-RTD Status Code: 9, details: ""
```

The workaround is to run "*/opt/aws/neuron/bin/neuron-cli reset*" to clear Neuron RTD states after all models are unloaded and server is shut down before restarting the model server.

Pipeline mode is not able to execute inferences requests in parallel

If you see that multiple executors in a neuron pipeline setup (one model compiled for more than one neuron-cores using *–neuroncore-pipeline-cores* option during compilation) are not running in parallel, please set the following MXNet's environment variables before inference to allow mxnet to execute the CPU ops in parallel. Otherwise it will be sequential and stall the executors.

`MXNET_CPU_WORKER_NTHREADS` is used to do that. Setting its value to `__subgraph_opt_neuroncore__` in the compiled model json will ensure that all the executors (threads) can be run in parallel.

Features only in MXNet-Neuron 1.5

- Shared memory for IFMaps transfer to neuron runtime (has higher performance compared to GRPC mode)
- Neuron profiling using MXNet

Features only in MXNet-Neuron 1.8

- Gluon API support
- Library mode neuron runtime

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.14 Apache MXNet Neuron (Incubating) Release Notes

Table of contents

- *Apache MXNet Neuron release [1.8.0.2.2.2.0]*
- *Apache MXNet Neuron release [1.8.0.2.1.5.0]*
- *Apache MXNet Neuron release [1.8.0.2.0.276.0]*
- *Apache MXNet Neuron release [1.8.0.2.0.271.0]*
- *[1.5.1.1.7.0.0]*

- *[1.5.1.1.6.5.0]*
- *[1.8.0.1.3.4.0]*
- *[1.5.1.1.6.1.0]*
- *[1.8.0.1.3.0.0]*
- *[1.8.0.1.2.1.0]*
- *[1.8.0.1.1.2.0]*
- *[1.5.1.1.4.x.x]*
- *[1.5.1.1.4.4.0]*
- *[1.5.1.1.3.8.0]*
- *[1.5.1.1.3.7.0]*
- *[1.5.1.1.3.2.0]*
- *[1.5.1.1.2.1.0]*
- *[1.5.1.1.1.88.0]*
- *[1.5.1.1.1.52.0]*
- *[1.5.1.1.1.1.0]*
- *[1.5.1.1.0.2101.0]*
- *[1.5.1.1.0.2093.0]*
- *[1.5.1.1.0.2033.0]*
- *[1.5.1.1.0.1900.0]*
- *[1.5.1.1.0.1596.0]*
- *[1.5.1.1.0.1498.0]*
- *[1.5.1.1.0.1401.0]*
- *[1.5.1.1.0.1325.0]*
- *[1.5.1.1.0.1349.0]*
- *[1.5.1.1.0.1325.0]*
- *[1.5.1.1.0.1260.0]*

This document lists the release notes for MXNet-Neuron framework.

10.14.1 Apache MXNet Neuron release [1.8.0.2.2.2.0]

Date: 03/25/2022

New in this release

- Added support for unloading models from a NeuronDevice by deleting the model instance in user application. Users can now call `del` in Python on an executor and to unload the model from a NeuronDevice (provided the deleted executor is the last executor pointing to the given model). This requires the latest `aws-mx-1.8` package from `https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl`.

Bug fixes

- Fixed a memory leak caused by stale unloaded models in NeuronDevice memory. For this fix to take effect please install aws-mx package from https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl along with the latest mx-neuron package.

10.14.2 Apache MXNet Neuron release [1.8.0.2.1.5.0]

Date: 01/20/2022

New in this release

- Added support of `mx_neuron.__version__` to get the build version of MXNet Neuron plugin

Bug fixes

- Fixed assertion errors when inference was completed with NaNs. The expected behavior is to complete inference successfully and warn the user that ``NaN``s were seen during the current inference.
- Fixed compile issue when individual output nodes have multiple output nodes. Because the output index was being dropped, fewer number of output feature maps were being considered and that caused failures during inference.

10.14.3 Apache MXNet Neuron release [1.8.0.2.0.276.0]

Date: 11/05/2021

- Updated Neuron Runtime (which is integrated within this package) to `libnrt 2.2.18.0` to fix a container issue that was preventing the use of containers when /dev/neuron0 was not present. See details here neuron-runtime-release-notes.

10.14.4 Apache MXNet Neuron release [1.8.0.2.0.271.0]

New in this release

- MXNet Neuron 1.8 now support Neuron Runtime 2.x (`libnrt.so` shared library) only.

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.
  - Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

- Introducing Flexible Execution Groups (FlexEG) feature. See *Flexible Execution Group (FlexEG) in Neuron-MXNet* application note.

Resolved Issues

- Fixed a bug that prevented compilation of gluon models with multiple cpu and neuron nodes.
- Added more debug logic to help with profiling of model load timing.

10.14.5 [1.5.1.1.7.0.0]

New in this release

- MXNet 1.5 enters maintenance mode. Please visit maintenance_mxnet_1_5 for more information.

Resolved Issues

- Minor bug fixes.

10.14.6 [1.5.1.1.6.5.0]

Date 08/12/2021

Summary

Minor bug fixes and enhancements for MXNet 1.5 Neuron.

10.14.7 [1.8.0.1.3.4.0]

Date 08/12/2021

Summary

Minor bug fixes and enhancements for MXNet 1.8 Neuron.

10.14.8 [1.5.1.1.6.1.0]

Date 07/02/2021

Summary

Minor bug fixes and enhancements for MXNet 1.5 Neuron.

10.14.9 [1.8.0.1.3.0.0]

Date 07/02/2021

Summary

Support for Autoloop, Cpredict API and minor bug fixes and enhancements for MXNet 1.8 Neuron.

Major New Features

- Added support for Autoloop feature for MXNet 1.8 Neuron.

Resolved Issues

- Added support for CPredict API.

10.14.10 [1.8.0.1.2.1.0]

Date 5/28/2021

Summary

Minor bug fixes and enhancements for MXNet 1.8 Neuron

Resolved Issues

- Added support for Neuron profiler

10.14.11 [1.8.0.1.1.2.0]

Date 4/30/2021

Summary

Initial release of Apache MXNet (Incubating) 1.8 for Neuron

Major New Features

- Gluon API and Neuron support for NLP BERT models
- Neuron is now a plugin
- Please note new API changes to support plugin mode: *Neuron Apache MXNet (Incubating) Compilation Python API*

10.14.12 [1.5.1.1.4.x.x]

Date 5/28/2021

Summary

- Minor enhancements.

10.14.13 [1.5.1.1.4.4.0]

Date 4/30/2021

Summary

- Resolve an issue with Neuron profiling.

Resolved Issues

- Issue: when Neuron profiling is enabled in MXNet-Neuron 1.5.1 (using NEURON_PROFILE=<dir>), and TensorBoard is used to read in the profiled data, user would see an error messsage "panic: runtime error: index out of range". This issue is resolved in this release.

10.14.14 [1.5.1.1.3.8.0]

Date 3/4/2021

Summary

Minor enhancements.

10.14.15 [1.5.1.1.3.7.0]

Date 2/24/2021

Summary

Fix for CVE-2021-3177.

10.14.16 [1.5.1.1.3.2.0]

Date 1/30/2021

Summary

Various minor improvements

10.14.17 [1.5.1.1.2.1.0]

Date 12/23/2020

Summary

Various minor improvements

10.14.18 [1.5.1.1.1.88.0]

Date 11/17/2020

Summary

This release includes the bug fix for MXNet Model Server not being able to clean up Neuron RTD states after model is unloaded (deleted) from model server.

Resolved Issues

- Issue: MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server.
  - Workaround for earlier versions: run "`/opt/aws/neuron/bin/neuron-cli reset`" to

  clear Neuron RTD states after all models are unloaded and server is shut down.

10.14.19 [1.5.1.1.1.52.0]

Date 09/22/2020

Summary

Various minor improvements.

Major New Features

Resolved Issues

- Issue: When first importing MXNet into python process and subprocess call is invoked, user may get an OSError exception "OSError: [Errno 14] Bad address" during subprocess call (see https://github.com/apache/incubator-mxnet/issues/13875 for more details). This issue is fixed with a mitigation patch from MXNet for Open-MP fork race conditions.
  - Workaround for earlier versions: Export KMP_INIT_AT_FORK=false before running python process.

10.14.20 [1.5.1.1.1.1.0]

Date 08/08/2020

Summary

Various minor improvements.

Major New Features

Resolved Issues

10.14.21 [1.5.1.1.0.2101.0]

Date 08/05/2020

Summary

Various minor improvements.

Major New Features

Resolved Issues

10.14.22 [1.5.1.1.0.2093.0]

Date 07/16/2020

Summary

This release contains a few bug fixes and user experience improvements.

Major New Features

Resolved Issues

- User can specify NEURONCORE_GROUP_SIZES without brackets (for example, "1,1,1,1"), as can be done in TensorFlow-Neuron and PyTorch-Neuron.
- Fixed a memory leak when inferring neuron subgraph properties
- Fixed a bug dealing with multi-input subgraphs

10.14.23 [1.5.1.1.0.2033.0]

Date 6/11/2020

Summary

- Added support for profiling during inference

Major New Features

- Profiling can now be enabled by specifying the profiling work directory using NEURON_PROFILE environment variable during inference. For an example of using profiling, see tensorboard-neuron. (Note that graph view of MXNet graph is not available via TensorBoard).

Resolved Issues

Known Issues and Limitations

Other Notes

10.14.24 [1.5.1.1.0.1900.0]

Date 5/11/2020

Summary

Improved support for shared-memory communication with Neuron-Runtime.

Major New Features

- Added support for the BERT-Base model (base: L-12 H-768 A-12), max sequence length 64 and batch size of 8.
- Improved security for usage of shared-memory for data transfer between framework and Neuron-Runtime
- Improved allocation and cleanup of shared-memory resource
- Improved container support by automatic falling back to GRPC data transfer if shared-memory cannot be allocated by Neuron-Runtime

Resolved Issues

- User is unable to allocate Neuron-Runtime shared-memory resource when using MXNet-Neuron in a container to communicate with Neuron-Runtime in another container. This is resolved by automatic falling back to GRPC data transfer if shared-memory cannot be allocated by Neuron-Runtime.
- Fixed issue where some large models could not be loaded on inferentia.

Known Issues and Limitations

Other Notes

10.14.25 [1.5.1.1.0.1596.0]

Date 3/26/2020

Summary

No major changes or fixes

Major New Features

Resolved Issues

Known Issues and Limitations

Other Notes

10.14.26 [1.5.1.1.0.1498.0]

Date 2/27/2020

Summary

No major changes or fixes.

Major New Features

Resolved Issues

The issue(s) below are resolved:

- Latest pip version 20.0.1 breaks installation of MXNet-Neuron pip wheel which has py2.py3 in the wheel name.

Known Issues and Limitations

- User is unable to allocate Neuron-Runtime shared-memory resource when using MXNet-Neuron in a container to communicate with Neuron-Runtime in another container. To work-around, please set environment variable NEURON_RTD_USE_SHM to 0.

Other Notes

10.14.27 [1.5.1.1.0.1401.0]

Date 1/27/2020

Summary

No major changes or fixes.

Major New Features

Resolved Issues

- The following issue is resolved when the latest multi-model-server with version >= 1.1.0 is used with MXNet-Neuron. You would still need to use "`/opt/aws/neuron/bin/neuron-cli reset`" to clear all Neuron RTD states after multi-model-server is exited:
  - Issue: MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server and previous workaround "`/opt/aws/neuron/bin/neuron-cli reset`" is unable to clear all Neuron RTD states.

Known Issues and Limitations

- Latest pip version 20.0.1 breaks installation of MXNet-Neuron pip wheel which has py2.py3 in the wheel name. This breaks all existing released versions. The error looks like:

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
ERROR: Could not find a version that satisfies the requirement mxnet-neuron (from␣
↪versions: none)
ERROR: No matching distribution found for mxnet-neuron
```

- Work around: install the older version of pip using "pip install pip==19.3.1".

Other Notes

10.14.28 [1.5.1.1.0.1325.0]

Date 12/1/2019

Summary

Major New Features

Resolved Issues

- Issue: Compiler flags cannot be passed to compiler during compile call. The fix: compiler flags can be passed to compiler during compile call using "flags" option followed by a list of flags.
- Issue: Advanced CPU fallback option is a way to attempt to improve the number of operators on Inferentia. The default is currently set to on, which may cause failures. The fix: This option is now off by default.

Known Issues and Limitations

- Issue: MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server and previous workaround "`/opt/aws/neuron/bin/neuron-cli reset`" is unable to clear all Neuron RTD states.
  - Workaround: run "`sudo systemctl restart neuron-rtd`" to clear Neuron RTD states after all models are unloaded and server is shut down.

Other Notes

10.14.29 [1.5.1.1.0.1349.0]

Date 12/20/2019

Summary

No major changes or fixes. Released with other Neuron packages.

10.14.30 [1.5.1.1.0.1325.0]

Date 12/1/2019

Summary

Major New Features

Resolved Issues

- Issue: Compiler flags cannot be passed to compiler during compile call. The fix: compiler flags can be passed to compiler during compile call using "flags" option followed by a list of flags.
- Issue: Advanced CPU fallback option is a way to attempt to improve the number of operators on Inferentia. The default is currently set to on, which may cause failures. The fix: This option is now off by default.

Known Issues and Limitations

- Issue: MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server and previous workaround "`/opt/aws/neuron/bin/neuron-cli reset`" is unable to clear all Neuron RTD states.
  - Workaround: run "`sudo systemctl restart neuron-rtd`" to clear Neuron RTD states after all models are unloaded and server is shut down.

Other Notes

10.14.31 [1.5.1.1.0.1260.0]

Date: 11/25/2019

Summary

This version is available only in released DLAMI v26.0 and is based on MXNet version 1.5.1. Please dlami-rn-known-issues to latest version.

Major new features

Resolved issues

Known issues and limitations

- Issue: Compiler flags cannot be passed to compiler during compile call.
- Issue: Advanced CPU fallback option is a way to attempt to improve the number of operators on Inferentia. The default is currently set to on, which may cause failures.
  - Workaround: explicitly turn it off by setting compile option op_by_op_compiler_retry to 0.
- Issue: Temporary files are put in current directory when debug is enabled.
  - Workaround: create a separate work directory and run the process from within the work directory
- Issue: MXNet Model Server is not able to clean up Neuron RTD states after model is unloaded (deleted) from model server.
  - Workaround: run "`/opt/aws/neuron/bin/neuron-cli reset`" to clear Neuron RTD states after all models are unloaded and server is shut down.
- Issue: MXNet 1.5.1 may return inconsistent node names for some operators when they are the primary outputs of a Neuron subgraph. This causes failures during inference.
  - Workaround : Use the `excl_node_names` compilation option to change the partitioning of the graph during compile so that these nodes are not the primary output of a neuron subgraph. See *Neuron Apache MXNet (Incubating) Compilation Python API*

```
compile_args = { 'excl_node_names': ["node_name_to_exclude"] }
```

Models Supported

The following models have successfully run on neuron-inferentia systems

1. Resnet50 V1/V2
2. Inception-V2/V3/V4
3. Parallel-WaveNet
4. Tacotron 2
5. WaveRNN

Other Notes

- Python versions supported:
  - 3.5, 3.6, 3.7
- Linux distribution supported:
  - Ubuntu 18, Amazon Linux 2

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

10.15 Neuron Apache MXNet (Incubating) Supported operators

To see a list of supported operators for MXNet, run the following command:

`neuron-cc list-operators --framework MXNET`

10.15.1 Neuron Compiler Release [1.6.13.0]

Added

```
amp_cast
amp_multicast
```

10.15.2 Neuron Compiler Release [1.4.1.0]

No changes

10.15.3 Neuron Compiler Release [1.4.0.0]

No changes

10.15.4 Neuron Compiler Release [1.3.0.0]

No changes

10.15.5 Neuron Compiler Release [1.2.7.0]

No changes

10.15.6 Neuron Compiler Release [1.2.2.0]

No changes

10.15.7 Neuron Compiler Release [1.2.0.0]

Added

```
Deconvolution
LayerNorm
Pad
SwapAxis
_contrib_arange_like
_contrib_interleaved_matmul_encdec_qk
_contrib_interleaved_matmul_encdec_valatt
_contrib_interleaved_matmul_selfatt_qk
_contrib_interleaved_matmul_selfatt_valatt
arctan
broadcast_like
cos
erf
pad
sin
slice_axis
```

10.15.8 Neuron Compiler Release [1.0.24045.0]

Added `_contrib_div_sqrt_dim`, `broadcast_axis`

10.15.9 Neuron Compiler Release [1.0.18001.0]

No changes

10.15.10 Neuron Compiler Release [1.0.17937.0]

No changes

10.15.11 Neuron Compiler Release [1.0.16861.0]

Removed `log` (Was erroneously reported as added in previous release.)

10.15.12 Neuron Compiler Release [1.0.15275]

Added `log`

10.15.13 Neuron Compiler Release [1.0.12696]

No changes

10.15.14 Neuron Compiler Release [1.0.9410]

No changes

10.15.15 Neuron Compiler Release [1.0.7878]

No changes

10.15.16 Neuron Compiler Release [1.0.6801]

No changes

10.15.17 Neuron Compiler Release [1.0.5939]

no changes

10.15.18 Neuron Compiler Release [1.0.5301]

no changes

10.15.19 Neuron Compiler Release [1.0.4680.0]

```
Activation
BatchNorm
Cast
Concat
Convolution
Convolution_v1
Dropout
Flatten
FullyConnected
LeakyReLU
Pooling
Pooling_v1
RNN
Reshape
SequenceMask
SliceChannel
Softmax
```

(continues on next page)

(continued from previous page)

```
UpSampling
__add_scalar__
__div_scalar__
__mul_scalar__
__pow_scalar__
__rdiv_scalar__
__rpow_scalar__
__rsub_scalar__
__sub_scalar__
_arange
_copy
_div_scalar
_equal_scalar
_full
_greater_equal_scalar
_greater_scalar
_lesser_equal_scalar
_lesser_scalar
_maximum
_maximum_scalar
_minimum
_minimum_scalar
_minus_scalar
_mul_scalar
_not_equal_scalar
_ones
_plus_scalar
_power_scalar
_rdiv_scalar
_rminus_scalar
_rnn_param_concat
_zeros
batch_dot
broadcast_add
broadcast_div
broadcast_equal
broadcast_greater
broadcast_greater_equal
broadcast_lesser
broadcast_lesser_equal
broadcast_maximum
broadcast_minimum
broadcast_mod
broadcast_mul
broadcast_not_equal
broadcast_sub
ceil
clip
concat
elemwise_add
elemwise_div
elemwise_mul
```

(continues on next page)

(continued from previous page)

```
elemwise_sub
exp
expand_dims
flatten
floor
gather_nd
log
log_softmax
max
mean
min
negative
ones_like
relu
repeat
reshape
reshape_like
reverse
rsqrt
sigmoid
slice
slice_like
softmax
split
sqrt
square
squeeze
stack
sum
tanh
tile
transpose
where
zeros_like
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

ELEVEN

NEURON RUNTIME

Neuron runtime consists of kernel driver and C/C++ libraries which provides APIs to access Inferentia and Trainium Neuron devices. The Neuron ML frameworks plugins for TensorFlow, PyTorch and Apache MXNet use the Neuron runtime to load and run models on the NeuronCores. Neuron runtime loads compiled deep learning models, also referred to as Neuron Executable File Format (NEFF) to the Neuron devices and is optimized for high-throughput and low-latency.

API Reference Guide

This document is relevant for: `Inf1`, `Trn1`

11.1 Neuron Runtime Configuration

Neuron Runtime is responsible for executing ML models on Neuron Devices. Neuron Runtime determines which NeuronCore will execute which model and how to execute it. Configuration of the Neuron Runtime is controlled through the use of Environment variables at the process level. By default, Neuron framework extensions will take care of Neuron Runtime configuration on the user's behalf. Explicit configurations are also possible when attempting to achieve a desired behavior.

This guide provides an overview of the different environment variables available to configure Neuron Runtime behavior.

Table 1: Environment Variables

Name	Description	Type	Expected Values	Default Value
`NEURON_RT_VISIBLE_CORES`	Range of specific NeuronCores needed by the process	Integer range (like 1-3)	Any value or range between 0 to Max NeuronCore in the system.	None
`NEURON_RT_NUM_CORES`	Number of NeuronCores required by the process.	Integer	A value from 1 to Max NeuronCore in the system.	0, which is interpretted as "all"
`NEURON_RT_LOG_LOCATION`	Runtime log location	string	console or syslog	console
`NEURON_RT_LOG_LEVEL`	Runtime log verbose level	string	ERROR, WARNING, INFO, DEBUG, TRACE	ERROR
`NEURON_RT_EXEC_TIMEOUT`	Timeout for execution in seconds	Integer	0 to INT_MAX	30
`NEURON_RT_VALIDATE_NEFF`	Validate NEFF contents before loading into accelerator	Boolean	TRUE or FALSE	FALSE

11.1.1 NeuronCore Allocation

Important: NEURONCORE_GROUP_SIZES is being deprecated, if your application is using NEURONCORE_GROUP_SIZES please see *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for more details.

By default, Neuron Runtime initializes all the cores present in the system and reserves them for the current process.

Note: Once a NeuronCore is reserved for a process, it cannot be used by another process at all, until the process reserving that NeuronCore is terminated.

Using NEURON_RT_VISIBLE_CORES

For parallel processing, NEURON_RT_VISIBLE_CORES can be used to control which NeuronCores each process would reserve. This variable is specified with a single NeuronCore index or an inclusive range value.

For example, if a process (myapp.py) requires one NeuronCore, then it can be started with NEURON_RT_VISIBLE_CORES=0 to limit the process to NeuronCore 0. For parallel processing, multiple process can be started (without any change to myapp.py code) with different NEURON_RT_VISIBLE_CORES values. Here is an example that runs myapp.py on inf1.xlarge in parallel across the four different NeuronCores available in the inf1.xlarge.

```
NEURON_RT_VISIBLE_CORES=0 myapp.py &
NEURON_RT_VISIBLE_CORES=1 myapp.py &
NEURON_RT_VISIBLE_CORES=2 myapp.py &
NEURON_RT_VISIBLE_CORES=3 myapp.py &
```

If myapp.py required 3 NeuronCores and was running on a inf1.6xlarge (16 NeuronCores maximum), the first instance of myapp.py could use NeuronCores 0-2, the next instance could use 3-5 and so on:

```
NEURON_RT_VISIBLE_CORES=0-2 myapp.py &
NEURON_RT_VISIBLE_CORES=3-5 myapp.py &
NEURON_RT_VISIBLE_CORES=6-8 myapp.py &
NEURON_RT_VISIBLE_CORES=9-11 myapp.py &
NEURON_RT_VISIBLE_CORES=12-14 myapp.py &
```

Using NEURON_RT_NUM_CORES

If NEURON_RT_NUM_CORES is set to a value between 1 and the maximum number of NeuronCores in the instance, Neuron Runtime will attempt to automatically reserve the number of free NeuronCores specified for the process. The difference between NEURON_RT_VISIBLE_CORES and NEURON_RT_NUM_CORES is that, NEURON_RT_VISIBLE_CORES specifies exact NeuronCores to allocate where as NEURON_RT_NUM_CORES specifies the number of NeuronCores needed and Neuron Runtime selects free NeuronCores.

Using the same example earlier where myapp.py needed 3 cores, but _which_ 3 cores was of no concern, the same application could be executed in parallel up to 5 times on an inf1.6xlarge (16 NeuronCore max):

```
NEURON_RT_NUM_CORES=3 myapp.py &
NEURON_RT_NUM_CORES=3 myapp.py &
NEURON_RT_NUM_CORES=3 myapp.py &
```

(continues on next page)

(continued from previous page)

```
NEURON_RT_NUM_CORES=3 myapp.py &
NEURON_RT_NUM_CORES=3 myapp.py &
```

Executing a 6th `NEURON_RT_NUM_CORES=3 myapp.py &` in the above example would fail as there is only a single NeuronCore still free.

Notes

1. Number of NeuronCores in a inferentia device is 4
2. Number of inferentia is depends on the instance size.
3. The NeuronCore index in NEURON_RT_VISIBLE_CORES starts from 0 and ends at (number of NeuronDevices * number of NeuronCores) - 1.
4. By default, `NEURON_RT_NUM_CORES` is set to `0`, which indicates to RT that all cores are to be used.
5. NEURON_RT_VISIBLE_CORES takes precedence over NEURON_RT_NUM_CORES. If specified, all cores within the range will be assigned to the owning process.

11.1.2 Logging and debug-ability

By default, Neuron Runtime logs to syslog with verbose level of *INFO* and only *ERROR* s are logged in console. The following code snippet shows ways to increase/decrease the log level.

```
NEURON_RT_LOG_LEVEL=INFO myapp.py         # Sets the log level for syslog and console to␣
↪INFO
NEURON_RT_LOG_LOCATION=console NEURON_RT_LOG_LEVEL=QUIET myapp.py    # Completely␣
↪disables console logging.
```

By default, Neuron Runtime expects the NeuronCore to complete execution of any model with in 2 seconds. If NeuronCore didnt complete the execution within 2 seconds then runtime would fail the execution with timeout error. Most of the models takes few milliseconds to complete so 2 seconds(2000 milliseconds) is more than adequate. However if your model is expected to run more than 2 seconds then you can increase the timeout with NEURON_RT_EXEC_TIMEOUT.

```
NEURON_RT_EXEC_TIMEOUT=5 myapp.py       # increases the timeout to 5 seconds
```

11.1.3 Checksum

To execute a model(NEFF), Neuron Runtime needs to load the NEFF file into NeuronCore and run. Neuron Runtime provides a way to do checksum validation on each NEFF file while loading to validate the file is not corrupted. This option is off by default to avoid performance penalty during model load time(~50%).

```
NEURON_RT_VALIDATE_HASH=true myapp1.py     # enables model checksum validation while␣
↪loading
NEURON_RT_VALIDATE_HASH=false myapp2.py    # disables(default) model checksum validation␣
↪while loading
```

This document is relevant for: `Inf1`, `Trn1`

11.2 Neuron Runtime Troubleshooting on Trn1

Table of contents

- *Memory Errors*
  - *Transient memory errors*
  - *Persistent memory errors*
  - *Failure to initialize Neuron*
  - *An application is trying to use more cores that are available on the instance*
- *EFA and Collective Communication Errors*
  - *Missing aws-neuronx-collectives package*
  - *Missing efa installer package.*
  - *EFA is not enabled in trn1.32xlarage*
  - *Communication timeout*
  - *Communication errors.*
  - *EFA Kernel messages (dmesg) after process termination.*
  - *Failure to find bootstrap interface*
  - *Name resolution failure*

11.2.1 Memory Errors

Transient memory errors

```
Uncorrectable memory error is detected on Neuron device: 5:1 metadata: 0x2. The error␣
↪might cause incorrect computational results and might affect training convergence.␣
↪Please
terminate and restart from the last checkpoint if the convergence is impacted.
```

Solution

Neuron detected a single uncorrectable bit flip in the device memory. The execution can continue but there is a possibility of a numerical error. If this is a concern, terminate and restart from the last known good check point.

Persistent memory errors

```
Uncorrectable memory error is detected on Neuron device: 5:1 metadata: 0x2. Failing␣
↪execution.
```

Solution

Multiple uncorrectable errors are detected during execution. The execution cannot continue. This is most likely caused by faulty hardware. Terminate and move to a different instance.

Failure to initialize Neuron

```
nd0 nc0 Timestamp program stop timeout (1000 ms)
nd0 nc0 Error while waiting for timestamp program to end on TPB eng 0
nd0 nc0 Failed to stop neuron core
nd0 nc0 Failed to end timestamp sync programs
TDRV not initialized
Failed to initialize devices, error:5
```

Solution

Previously executed application left Neuron devices in running state. Reset Neuron devices but reloading Neuron Driver. Note, this is a temporary workaround, future versions of Neuron will reset running devices automatically.

```
sudo rmmod neuron; sudo modprobe neuron
```

An application is trying to use more cores that are available on the instance

```
Could not open the nd1
```

Solution

Use properly sized instance. trn1.32xlarge has 32 Neuron Cores, trn1.2xlarge has 2 Neuron Cores.

11.2.2 EFA and Collective Communication Errors

Missing aws-neuronx-collectives package

aws-neuronx-collectives package is required to execute Collective Communication on a single instance and across multiple instances.

```
NCCL init error: Error opening libnccom.so, cannot use collective operations! Please set␣
↪LD_LIBRARY_PATH to library location. Error: libnccom.so: cannot open shared object
file: No such file or directory
Please make sure to install correct version of aws-neuronx-collectives; for detailed␣
↪installation instructions visit Neuron documentation
```

Solution

Install aws-neuornx-collectives package. If the installation used non-default destination set LD_LIBRARY_PATH.

Missing efa installer package.

efa-installer package is required to execute Collective Communication across multiple instances.

```
Unable to run multi-instance workload.  Ofi plugin is not installed or EFA is not enabled
```

Solution

Follow the directions to install efa-installer package. Make sure to add the path to to libfabric library to LD_LIBRARY_PATH

EFA is not enabled in trn1.32xlarage

EFA is used as a transport for Collective Communication among multiple instances. EFA must be enabled on the instances used for multi-node training.

```
OFI plugin initNet() failed is EFA enabled?
```

Solution

Confirm that EFA is enabled by running lspci command and making sure there are eight EFA devices. For example:

```
[ec2-user@ip-10-0-13-247 ~]$ lspci -tv
-+-[0000:a0]-+-00.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-01.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-19.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1a.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1b.0  Amazon.com, Inc. NeuronDevice
 |           +-1c.0  Amazon.com, Inc. NeuronDevice
 |           +-1d.0  Amazon.com, Inc. NeuronDevice
 |           +-1e.0  Amazon.com, Inc. NeuronDevice
```

(continues on next page)

(continued from previous page)

```
 |           \-1f.0  Amazon.com, Inc. NVMe SSD Controller
 +-[0000:90]-+-00.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-01.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-19.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1a.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1b.0  Amazon.com, Inc. NeuronDevice
 |           +-1c.0  Amazon.com, Inc. NeuronDevice
 |           +-1d.0  Amazon.com, Inc. NeuronDevice
 |           +-1e.0  Amazon.com, Inc. NeuronDevice
 |           \-1f.0  Amazon.com, Inc. NVMe SSD Controller
 +-[0000:20]-+-00.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-01.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-19.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1a.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1b.0  Amazon.com, Inc. NeuronDevice
 |           +-1c.0  Amazon.com, Inc. NeuronDevice
 |           +-1d.0  Amazon.com, Inc. NeuronDevice
 |           +-1e.0  Amazon.com, Inc. NeuronDevice
 |           \-1f.0  Amazon.com, Inc. NVMe SSD Controller
 +-[0000:10]-+-00.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-01.0  Amazon.com, Inc. Elastic Network Adapter (ENA)
 |           +-19.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1a.0  Amazon.com, Inc. Elastic Fabric Adapter (EFA)
 |           +-1b.0  Amazon.com, Inc. NeuronDevice
 |           +-1c.0  Amazon.com, Inc. NeuronDevice
 |           +-1d.0  Amazon.com, Inc. NeuronDevice
 |           +-1e.0  Amazon.com, Inc. NeuronDevice
 |           \-1f.0  Amazon.com, Inc. NVMe SSD Controller
 \-[0000:00]-+-00.0  Intel Corporation 440FX - 82441FX PMC [Natoma]
             +-01.0  Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
             +-01.3  Intel Corporation 82371AB/EB/MB PIIX4 ACPI
             +-03.0  Amazon.com, Inc. Device 1111
             +-04.0  Amazon.com, Inc. NVMe EBS Controller
             \-1f.0  Amazon.com, Inc. NVMe EBS Controller
```

Launch instances with EFA enabled and try again. If not planning to use the instances for multi-node training or running on trn1.2xlarge, this error message can be ignored.

Communication timeout

Ranks exchange information during NEFF loading and before the start of the execution. The loading/execution cannot move forward until all ranks are ready.

```
Timeout waiting for RX (waited 120 sec) - retrying
```

```
Timeout waiting for incoming connection (waited 120 sec) - retrying
```

```
Connect to localhost:33666 failed - retrying
```

Solution

The communication timeouts are not fatal. The ranks will continue waiting forever. In most case the timeouts are caused by one of the ranks getting delayed, usually be recompilation of a graph. The execution is resumed after the graph is compiled (might take significant amount of time). It is possible to determine if compilation is in progress by checking the logs on all nodes.

Communication timeouts might also indicate that one of the nodes or ranks is hang. If that is the case, terminate the run and restart from the last known good check point.

Communication errors.

```
RX, connection closed by remote peer
```

There could be other similar messages indicating that ranks failed to communicate.

Solution

One of the ranks or nodes encountered a problem and terminated. Terminate the run and restart from the last known check point.

EFA Kernel messages (dmesg) after process termination.

```
[298850.502143] neuron:npid_detach: neuron:npid_detach: pid=90193, slot=0
[298850.919248] efa 0000:a0:1a.0 rdmap160s26: Failed to process command DEREG_MR (opcode␣
↪8) comp_status 7 err -22
```

Solution

When a process that executed Collective Communication terminates it deregisters buffers that were registered with the networking stack. There is a race condition because the Neuron driver deregisters buffers owned by terminating process as part of the memory cleanup. The error is benign and will be removed in the future releases.

Failure to find bootstrap interface

```
No interface found in the same subnet as remote address fe80::1461:22ff:fe33:b471<45015>
No usable listening interface found
```

Solution

Bootstrap code incorrectly trying to use link-local IPv6 address for communication. This error will be fixed in the next Neuron release. In the meantime, as a workaround, disable IPv6 on the instances.

```
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1
```

Name resolution failure

```
WARN Invalid NCCL_COMM_ID [compute1-st-kaena-training-0-1.pcluster-trn1-24-pdx80-2n.
↪pcluster:41211], please use format: <ipv4>:<port> or [<ipv6>]:<port>
```

Solution

Verify that the name can be resolved by DNS by using nslookup or dig. Currently released version fails to resolve FQDN longer than 63 characters. This error will be fixed in the upcoming Neuron release. In the mean time use shorter names to ensure that FQDN length does not exceed the maximum of 63 characters.

This document is relevant for: `Inf1`

11.3 Neuron Runtime Troubleshooting on Inf1

This document aims to provide more information on how to fix issues you might encounter while using the Neuron Runtime 2.x or above. For each issue we will provide an explanation of what happened and what can potentially correct the issue.

If your issue is not listed below or you have a more nuanced problem, contact us via issues posted to this repo, the AWS Neuron developer forum, or through AWS support.

Table of contents

- *Neuron Driver installation fails*
  - *Solution*
- *Application fails to start*
  - *Solution*
- *Neuron Core is in use*
  - *Solution*
- *Unsupported NEFF Version*
  - *Solution*
- *Insufficient Memory*
  - *Solution*
- *Insufficient number of NeuronCores*
  - *Solution*
- *Numerical Error*
  - *Solution*

11.3.1 Neuron Driver installation fails

aws-neuron-dkms is a driver package which needs to be compiled during installation. The compilation requires kernel headers for the instance's kernel. `uname -r` can be used to find kernel version in the instance. In some cases, the installed kernel headers might be newer than the instance's kernel itself.

Please look at the aws-neuron-dkms installation log for message like the following:

```
Building for 4.14.193-149.317.amzn2.x86_64
Module build for kernel 4.14.193-149.317.amzn2.x86_64 was skipped since the
kernel headers for this kernel does not seem to be installed.
```

If installation log is not available, check whether the module is loaded.

```
$ lsmod | grep neuron
```

If the above has no output then that means `aws-neuron-dkms` installation is failed.

Solution

1. Stop all applications using the NeuronCores.
2. Uninstall aws-neuron-dkms `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
3. Install kernel headers for the current kernel `sudo apt install -y linux-headers-$(uname -r)` or `sudo yum install -y kernel-devel-$(uname -r) kernel-headers-$(uname -r)`
4. Install aws-neuron-dkms `sudo apt install aws-neuron-dkms` or `sudo yum install aws-neuron-dkms`

11.3.2 Application fails to start

Neuron Runtime requires Neuron Driver(aws-neuron-dkms package) to access Neuron devices. If the driver is not installed then Neuron Runtime wont able to access the Neuron devices and will fail with an error message in console and syslog.

If `aws-neuron-dkms` is not installed then the error message will be like the following:

```
2021-Aug-11 18:38:27.0917 13713:13713 ERROR   NRT:nrt_init      Unable to determine␣
↪Neuron Driver version. Please check aws-neuron-dkms package is installed.
```

If `aws-neuron-dkms` is installed but does not support the latest runtime then the error message will be like the following:

```
2021-Aug-11 19:18:21.0661 24616:24616 ERROR   NRT:nrt_init      This runtime requires␣
↪Neuron Driver version 2.0 or greater. Please upgrade aws-neuron-dkms package.
```

When using any supported framework from Neuron SDK version 2.5.0 and Neuron Driver (aws-neuron-dkms) versions 2.4 or older, Neuron Runtime will return the following error message:

```
2022-Dec-01 09:34:12.0559   138:138   ERROR   HAL:aws_hal_tpb_pooling_write_profile     ␣
↪ failed programming the engine
```

Solution

Please follow the installation steps in *Setup Guide* to install `aws-neuronx-dkms`.

11.3.3 Neuron Core is in use

A Neuron Core cant be shared between two applications. If an application started using a Neuron Core all other applications trying to use the NeuronCore would fail during runtime initialization with the following message in the console and in syslog:

```
2021-Aug-27 23:22:12.0323 28078:28078 ERROR   NRT:nrt_allocate_neuron_cores
↪ NeuronCore(s) not available - Requested:nc1-nc1 Available:0
```

Solution

Terminate the the process using NeuronCore and then try launching the application again.

11.3.4 Unsupported NEFF Version

While loading a model(NEFF), Neuron Runtime checks the version compatibility. If the version the NEFF is incompatible with Runtime then it would fail the model load with following error message:

```
NEFF version mismatch supported: 1.1 received: 2.0
```

Solution

Use compatible versions of Neuron Compiler and Runtime. Updating to the latest version of both Neuron Compiler and Neuron Runtime is the simplest solution. If updating one of the two is not an option, please refer to the neuron-runtime-release-notes of the Neuron Runtime to determine NEFF version support.

11.3.5 Insufficient Memory

While loading a model(NEFF), Neuron Runtime reserves both device and host memory for storing weights, ifmap and ofmap of the Model. The memory consumption of each model is different. If Neuron Runtime is unable to allocate memory then the model load would fail with the following message in syslog

```
kernel: [XXXXX] neuron:mc_alloc: device mempool [0:0] total 1073741568 occupied
↪960539030 needed 1272 available 768
```

Solution

As the error is contextual to what's going on with your instance, the exact next step is unclear. Try unloading some of the loaded models which will free up device DRAM space. If this is still a problem, moving to a larger Inf1 instance size with additional NeuronCores may help.

11.3.6 Insufficient number of NeuronCores

The NEFF requires more NeuronCores than available on the instance.

Check for error messages in syslog similar to:

```
NRT:  26638:26638 ERROR  TDRV:db_vtpb_get_mla_and_tpb                Could not find VNC␣
↪id n
NRT:  26638:26638 ERROR  NMGR:dlr_kelf_stage                         Failed to create␣
↪shared io
NRT:  26638:26638 ERROR  NMGR:stage_kelf_models                      Failed to stage␣
↪graph: kelf-a.json to NeuronCore
NRT:  26638:26638 ERROR  NMGR:kmgr_load_nn_post_metrics              Failed to load NN:␣
↪xxxxxxx, err: 2
```

Solution

The NeuronCores may be in use by models you are not actively using. Ensure you've unloaded models you're not using and terminated unused applications. If this is still a problem, moving to a larger Inf1 instance size with additional NeuronCores may help.

11.3.7 Numerical Error

Neuron Devices will detect any NaN generated during execution and report it. If Neuron Runtime sees NaNs are generated then it would fail the execution request with Numerical Error with the following message:

```
nrtd[nnnnn]: ....  Error notifications found on NC .... INFER_ERROR_SUBTYPE_NUMERICAL
```

Solution

This usually an indication of either error in the model or error in the input.

Report issue to Neuron by posting the relevant details on GitHub issues.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

11.4 Neuron runtime FAQ

Table of Contents

- *Where can I find information about Neuron Runtime 2.x (`libnrt.so`)*
- *What will happen if I will upgrade Neuron Framework without upgrading latest kernel mode driver?*
- *Do I need to recompile my model to use the Runtime Library?*
- *Do I need to change my application launch command?*
- *How do I restart/start/stop the Neuron Runtime?*
- *How do I know which runtimes are associated with which Neuron Device(s)?*
- *What about RedHat or other versions of Linux and Windows?*
- *How can I take advantage of multiple NeuronCores to run multipleinferences in parallel?*

11.4.1 Where can I find information about Neuron Runtime 2.x (`libnrt.so`)

See *Introducing Neuron Runtime 2.x (libnrt.so)* for detailed information about Neuron Runtime 2.x (`libnrt.so`).

11.4.2 What will happen if I will upgrade Neuron Framework without upgrading latest kernel mode driver?

Application start would fail with the following error message: .. code:: bash

> 2021-Aug-11 19:18:21.0661 24616:24616 ERROR NRT:nrt_init This runtime requires Neuron Driver version 2.0 or greater. Please upgrade aws-neuron-dkms package.

11.4.3 Do I need to recompile my model to use the Runtime Library?

No. Runtime 2.x supports all the models compiled with Neuron Compiler 1.x.

11.4.4 Do I need to change my application launch command?

No.

11.4.5 How do I restart/start/stop the Neuron Runtime?

Since Neuron Runtime is a library, starting/stopping application would result in starting/stopping the Neuron Runtime.

11.4.6 How do I know which runtimes are associated with which Neuron Device(s)?

neuron-ls and *neuron-top* can be used to find out applications using Neuron Devices.

11.4.7 What about RedHat or other versions of Linux and Windows?

We don't officially support it yet.

11.4.8 How can I take advantage of multiple NeuronCores to run multipleinferences in parallel?

Examples of this for TensorFlow and MXNet are found here and here.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

11.5 Neuron Runtime Release Notes

Neuron Runtime consists of a kernel mode driver and C/C++ libraries which provides APIs to access Neuron Devices. The runtime itself (libnrt.so) is integrated into the ML frameworks for simplicity of deployment. The Neuron Runtime supports training models and executing inference on the Neuron Cores.

Table of contents

- *Known issues*
- *NEFF Support Table:*
- *Neuron Runtime Library [2.10.18.0]*
- *Neuron Runtime Library [2.10.15.0]*
- *Neuron Runtime Library [2.9.64.0]*
- *Neuron Runtime 2.x (`libnrt.so`) release [2.2.51.0]*
- *Neuron Runtime 2.x (`libnrt.so`) release [2.2.31.0]*
- *Neuron Runtime 2.x (`libnrt.so`) release [2.2.18.0]*
- *Neuron Runtime 2.x (`libnrt.so`) release [2.2.15.0]*

11.5.1 Known issues

Updated : 04/29/2022

- In rare cases of multi-process applications running under heavy stress a model load failure my occur. This may require reloading of the Neuron Driver as a workaround.

11.5.2 NEFF Support Table:

Use this table to determine the version of Runtime that will support the version of NEFF you are using. NEFF version is determined by the version of the Neuron Compiler.

NEFF Version	Runtime Version Range	Notes
0.6	*	All versions of RT support NEFF 0.6
1.0	>= 1.0.6905.0	Starting support for 1.0 NEFFs
2.0	>= 1.6.5.0	Starting support for 2.0 NEFFs

11.5.3 Neuron Runtime Library [2.10.18.0]

Date: 11/07/2022

New in this release

- Minor bug fixes and enhancements.

11.5.4 Neuron Runtime Library [2.10.15.0]

Date: 10/26/2022

New in this release

- Changed default runtime behavior to reset NeuronCores when initializing applications. With this change, the reseting of the Neuron Driver after application crash is no longer necessary. The new reset functionality is controled by setting environment variable: `NEURON_RT_RESET_CORES`, see *Neuron Runtime Configuration* for more information.

Bug fixes

- Fixed a bug where Stochastic Rounding was not being set for collective communication operators
- Fixed an issue with triggering DMA for large tensors
- Increased default execution timeout to 30 seconds
- Fixed IOQ resetting queue to incorrect ring id value
- Updated the Neuron driver for more reliable behavior of driver device reset. Driver no longer busy waits on reset or gets stuck waiting on reset, which caused kernel taints or caused driver unload attempts to fail.
- Fixed a bug the prevented collective communication over tensors larger than 2GB
- Fixed a bug that caused intermittent memory corruption when unloading a model
- Fixed a bug that caused the exhausting of EFA memory registration pool after multiple model reloads.

11.5.5 Neuron Runtime Library [2.9.64.0]

Date: 10/10/2022

This release specifically adds support for training workloads on one or more EC2 TRN1 instances.

Required Neuron Driver Version: 2.5 or newer

New in this release

- Broke out runtime into a separate package called aws-neuronx-runtime-lib.
- Added RUNPATH for discovery of libnrt.so, can be overridden with LD_LIBRARY_PATH.
- Added support for multiple collective compute operations, e.g. All-Reduce, Reduce-Scatter, All-Gather.
- Added Send/Recv operation support
- Added support for using multiple DMA engines with single pseudo embedding update instruction.
- Changed instruction buffer alignment to 32K.
- Reduced memory required during NEFF swapping.
- Enabled notifications for send/recv collectives operations.
- Added trace apis in support of execution profiling.
- Added support for TPB reset (default: off).
- Added version checking for libnccom (aws-neuronx-collectives).
- Added new runtime version API.
- Added 8-channel support for Trn1.
- Improved debug outputs.
- Added support for write combining on BAR4.
- Increased default execution timeout from 2 seconds to 30 seconds.
- Improved handling of zero-sized tensors

11.5.6 Neuron Runtime 2.x (`libnrt.so`) release [2.2.51.0]

Date: 03/25/2022

- Fixed an invalid memory access that could occur when unloading models.
- Reduced severity of logging for numerical errors from ERROR to WARN.
- Improved handling of models with numerous CPU operations to avoid inference failure due to memory exhaustion.

11.5.7 Neuron Runtime 2.x (`libnrt.so`) release [2.2.31.0]

Date: 01/20/2022

New in the release

- Changed error notifications from `WARN` to `ERROR` in cases when the causing problem is non-recoverable.
- Changed handling of inference timeouts (`NERR_TIMEOUT`) to avoid failure when the timeout is related to a software thread scheduling conflict.

Bug fixes

- Increased the number of data queues in Neuron Runtime 2.x to match what was previously used in Neuron Runtime 1.x. The use of fewer number of data queues in Neuron Runtime 2.x was leading to crashes in a limited number of models.
- Fixed the way Neuron Runtime 2.x updates the inference end timestamp. Previously, Neuron Runtime 2.x update of the inference end timestamp would have lead to a negative latency statistics in neuron-monitor with certain models.

11.5.8 Neuron Runtime 2.x (`libnrt.so`) release [2.2.18.0]

Date: 11/05/2021

- Resolved an issue that affect the use of Neuron within container. In previous Neuron Runtime release (libnrt.so.2.2.15.0), when /dev/neuron0 was not used by the application, Neuron Runtime attempted and failed to initialize /dev/neuron0 because user didn't pass /dev/neuron0 to the container. this Neuron Runtime release (`libnrt.so.2.2.18.0`) allows customers to launch containers with specific NeuronDevices other than /dev/neuron0.

11.5.9 Neuron Runtime 2.x (`libnrt.so`) release [2.2.15.0]

Date: 10/27/2021

New in this release

- *First release of Neuron Runtime 2.x* - In this release we are introducing Neuron Runtime 2.x which is a shared library named (`libnrt.so`) and replacing Neuron Runtime 1.x server (`neruon-rtd`) . Upgrading to `libnrt.so` improves throughput and latency, simplifies Neuron installation and upgrade process, introduces new capabilities for allocating NeuronCores to applications, streamlines container creation, and deprecates tools that are no longer needed. The new library-based runtime (`libnrt.so`) is integrated into Neuron's ML Frameworks (with the exception of MXNet 1.5) and Neuron Tools packages directly - users no longer need to install/deploy the `aws-neuron-runtime`package.

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.

- Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

11.6 Neuron Driver Release Notes

Table of contents

- *Known issues*
- *Neuron Driver release [2.6.26.0]*
- *Neuron Driver release [2.5.38.0]*
- *Neuron Driver release [2.3.26.0]*
- *Neuron Driver release [2.3.11.0]*
- *Neuron Driver release [2.3.3.0]*
- *Neuron Driver release [2.2.14.0]*
- *Neuron Driver release [2.2.13.0]*
- *Neuron Driver release [2.2.6.0]*
- *Neuron Driver release [2.1]*

11.6.1 Known issues

Updated : 04/29/2022

- In rare cases of multi-process applications running under heavy stress a model load failure my occur. This may require reloading of the Neuron Driver as a workaround.

11.6.2 Neuron Driver release [2.6.26.0]

Date: 11/07/2022

New in this release

- Minor bug fixes and improvements.

11.6.3 Neuron Driver release [2.5.38.0]

Neuron Driver now supports INF1 and TRN1 EC2 instance types. Name of the driver package changed from aws-neuron-dkms to aws-neuronx-dkms. Please remove the older driver package before installing the newest one.

Date: 10/10/2022

New in this release

- Support added for EC2 Trn1 instance types and ML training workloads.
- Added missing GPL2 LICENSE file.
- Changed package name to aws-neuronx-dkms (was previously minus the 'x').
- Security Update – blocked user space access to control registers and DMA control queues intended to be used by the Neuron Driver only.
- Added support for DMA Aborts to avoid hangs.
- Added support for TPB Reset.
- Added sysfs entries for triggering resets and reading core counts.
- Added write combining on BAR4.
- Added PCI Device ID update as part of install.
- Added handling for known duplicate device id error.

Bug Fixes

- Fixed a null pointer free scenario.
- Fixed installation issue related to install without internet connectivity.

11.6.4 Neuron Driver release [2.3.26.0]

Date: 08/02/2022

Bug Fixes

- Security Update: Blocked user space access to control registers and DMA control queues intended to be used by the Neuron Driver only. Recommending upgrade to all customers.

11.6.5 Neuron Driver release [2.3.11.0]

Date: 05/27/2022

New in this release

- This driver is required to support future releases of the Neuron Runtime. Included in the release is both a bug fix to avoid a kernel crash scenario and an increased compatibility range to ensure compatibility with future versions of Neuron Runtime.

Bug Fixes

- Correction to huge aligned memory allocation/freeing logic that was previously susceptible to crashes in the kernel. The crash would bring down the OS. Recommending upgrade to all customers.

11.6.6 Neuron Driver release [2.3.3.0]

Date: 04/29/2022

New in this release

- Minor performance improvements on inference and loading of models.

Bug Fixes

- Reduced Host CPU usage when reading `hw_counters` metric from neuron-monitor
- Minor bug fixes.

11.6.7 Neuron Driver release [2.2.14.0]

Date: 03/25/2022

New in this release

- Minor updates

11.6.8 Neuron Driver release [2.2.13.0]

Date: 01/20/2022

New in this release

- Minor updates

11.6.9 Neuron Driver release [2.2.6.0]

Date: 10/27/2021

New in this release

- Memory improvements made to ensure all allocations are made with 4K alignments.

Resolved issues

- No longer delays 1s per NeuronDevice when closing Neuron Tools applications.
- Fixes a Ubuntu 20 build issue

11.6.10 Neuron Driver release [2.1]

- Support is added for Neuron Runtime 2.x (`libnrt.so`).
- Support for previous releases of Neuron Runtime 1.x is continued with Driver 2.x releases.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Trn1`

11.7 Neuron Collectives Release Notes

Neuron Collectives refers to a set of libraries used to support collective compute operations within the Neuron SDK. The collectives support is delivered via the aws-neuronx-collectives package and includes a pre-built version of the OFI plugin required for use of collectives with Elastic Fabric Adapter (EFA).

Table of contents

- *Neuron Collectives [2.10.20.0]*
- *Neuron Collectives [2.9.86.0]*

11.7.1 Neuron Collectives [2.10.20.0]

Date: 10/10/2022

New in this release

- Improved logging to appear similar in style to Neuron Runtime

Bug Fixes

- Fixed memory registration to support 2GB+ sizes
- Fixed association of network devices to channels (removes previous hard-coding).

11.7.2 Neuron Collectives [2.9.86.0]

Date: 10/10/2022

New in this release

- Added support for All-Reduce, Reduce-Scatter, All-Gather, and Send/Recv operations.

This document is relevant for: `Trn1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWELVE

NEURON COMPILER

The Neuron Compiler accepts Machine Learning models in various formats (TensorFlow, MXNet, PyTorch, XLA HLO) and optimizes them to run on Neuron devices.

The Neuron compiler is invoked within the ML framework, where ML models are sent to the compiler by the Neuron Framework plugin. The resulting compiler artifact is called a NEFF file (Neuron Executable File Format) that in turn is loaded by the Neuron runtime to the Neuron device.

Neuron Compiler for Trn1

API Reference Guide

This document is relevant for: `Trn1`

12.1 Neuron Compiler CLI Reference Guide (`neuronx-cc`)

This document describes the command line interface of the Neuron Compiler.

This reference is not relevant for applications that run the Neuron Compiler from within a machine learning framework (*PyTorch-Neuron* for example) since these options are passed from the framework directly to the compiler. Using the compiler command line may be desirable for applications that do not use a framework or customize existing frameworks. It is also possible to specify compiler options within the framework which will forward these options to the compiler using *NEURON_CC_FLAGS*.

12.1.1 Usage

Optional parameters are shown in square brackets.

Neuron Compiler Command-Line Interface

`neuronx-cc` `<command> [parameters]`

Common parameters for the Neuron CLI:

- `--verbose`: Specify the level of output produced by the compiler. (Default: `warning`)

  Valid values:

  - `info`: Informational messages regarding the progress of model compilation (written to stdout).

  - warning: Diagnostic messages that report model code that is not inherently erroneous but may be risky or suggest there may have been an error (written to stderr).
  - error: The compiler detected a condition causing it not complete the compilation successfully (written to stderr).
  - critical: The compiler encountered an unrecoverable error terminates immediately (written to stderr).
  - debug: Extensive information regarding the compiler's internal execution phases (written to stdout).
- --help: Display a usage message of compiler options.

  Use neuronx-cc <command> --help for information on a specific command.

Available Commands:

- compile
- list-operators

neuronx-cc compile [parameters]

Compile a model for use on the AWS Machine Learning Accelerator.

```
neuronx-cc compile <model_files>
--framework <framework_name>
--target <instance_family>
[--auto-cast <cast_mode>]
[--auto-cast-type <data_type>]
[--model-type <model>]
[--enable-fast-context-switch>]
[--enable-fast-loading-neuron-binaries]
[--logfile <filename>]
[--neuroncore-pipeline-cores <count>]
[--output <filename>]
```

Compile Parameters:

- <model_files>: Input containing model specification.

  The number of arguments required varies between frameworks:

  - **XLA**: A local filename of a HLO file (hlo.pb) generated via XLA. See hlo.proto for the .proto description and inspect-compiled-programs for more information on how to generate such files.
- --framework: Framework used to generate training model.

  Valid values:

  - XLA
- --target: Name of the Neuron instance family on which the compiled model will be run.

  Valid values:

  - trn1
- --model-type: Permit the compiler to attempt model-specific optimizations based upon type of model being compiled. (Default: generic)

  Valid values:

  - generic

  - `transformer` (wikipedia)

- `--auto-cast`: Controls how the compiler makes tradeoffs between performance and accuracy for FP32 operations. (Default: `matmult`)

  Valid values:

  - `matmul`: Only cast FP32 operations that use the Neuron matrix-multiplication engine.
  - `all`: Cast all FP32 operations to achieve highest performance. This option can potentially lower precision/accuracy.
  - `none`: Leave all data types as defined in the model. Do not apply auto-casting data type optimizations.

  A more complete discussion on how to use this option and its arguments is in *Mixed Precision and Performance-accuracy Tuning for Training*.

  Note: If the `--auto-cast` option is specified, the `--auto-cast-type` compiler flag can be optionally set to define which lower-precision data type the compiler should use.

- `--auto-cast-type`: When auto-cast mode is enabled, cast the FP32 operators to the lower-precision data type specified by this option. (Default: `bf16`)

  Valid values:

  - `bf16`: Cast the FP32 operations selected via the `--auto-cast` option to BF16 to achieve highest performance and preserve dynamic range.
  - `fp16`: Cast the FP32 operations selected via the `--auto-cast` option to FP16 to achieve improved performance relative to FP32 and increased precision relative to BF16.
  - `tf32`: Cast the FP32 operations selected via the `--auto-cast` option to TensorFloat-32.

  Note: If multiple competing options are specified then the option later in the command line will supercede previous options.

- `--enable-fast-context-switch`: Optimize for faster model switching rather than execution latency.

  This option will defer loading some weight constants until the start of model execution. This results in overall faster system performance when your application switches between models frequently on the same Neuron Core (or set of cores).

- `--enable-fast-loading-neuron-binaries`: Save the compilation output file in an uncompressed format.

  This creates executable files which are larger in size but faster for the Neuron Runtime to load into memory during model execution.

- `--logfile`: Filename where compiler writes log messages. (Default: "log-neuron-cc.txt").
- `--neuroncore-pipeline-cores`: Number of Neuron Cores to be used in "NeuronCore Pipeline" mode for low-latency inference. (Default: 1)

  Note: This is not used to define the number of Neuron Cores to be used in a data parallel deployment (i.e., the same model on multiple Neuron Cores). That is a Runtime/Framework configuration choice. Refer to *Parallel Execution using NEURON_RT_NUM_CORES* for more details.

- `--output`: Filename where compilation output (NEFF archive) will be recorded. (Default: "file.neff")

Example:

Compiling an XLA HLO:

```
neuronx-cc compile bert-model.hlo --framework XLA --target trn1 --model-
↪type transformer --output bert.neff
```

neuronx-cc `list-operators [parameters]`

Returns a newline ('\n') separated list of operators supported by the Neuron Compiler.

```
neuronx-cc list-operators
--framework <value>
```

List-Operators Parameters:

- `--framework`: Framework in which the operators were registered.

  Valid values:

  - `XLA`: Operator names will be formatted according to the value used by XLA compiler in XlaBuilder.

Example:

```
neuronx-cc list-operators -framework XLA
...
```

Exit Statuses:

- **0**: Compilation succeeded
- **<>0**: An error occurred during compilation.

This document is relevant for: `Trn1`

Developer Guide

This document is relevant for: `Trn1`

12.2 Mixed Precision and Performance-accuracy Tuning (`neuronx-cc`)

Table of contents

- *Overview*
- *Neuron Hardware*
- *Performance-accuracy tradeoffs*
- *What is the difference between Data Types?*
- *Should I downcast operations to smaller Data Types?*

12.2.1 Overview

The Neuron Compiler supports machine learning models with FP32, TF32, FP16 and BF16 (Bfloat16) tensors and operators. The Neuron hardware supports a mix of 32, 16, and 8 bit datatypes. This guide explains how to apply the available auto-cast methods and their performance / accuracy trade-offs when compiling a model with Neuron.

Note: Neuron Compiler support for INT8 is planned for a future Neuron SDK release. See Neuron Compiler: Enable Neuron INT8 support for details.

12.2.2 Neuron Hardware

The Neuron v2 hardware supports matrix multiplication using FP16, BF16, TF32, and FP32 on its matrix multiply ("matmult") engine, and accumulations using FP32. Operators such as activations or vector operations are supported using FP32, TF32, FP16, and BF16. Supporting FP16 and BF16 allows Neuron to have significantly higher performance than executing everything as FP32.

12.2.3 Performance-accuracy tradeoffs

By default, the Neuron Compiler will **automatically cast FP32 matrix multiplication operations to BF16**. The remaining operations are performed in the data type specified by the model. The Neuron Compiler provides CLI options that direct the compiler to cast to other data types, thereby giving the ability to choose an accuracy-to-performance tradeoff in model execution. Deciding what CLI settings to use will be application specific and may require some experimentation. See *Neuron Compiler CLI Reference Guide* for details.

12.2.4 What is the difference between Data Types?

The NeuronCore v2 support multiple data types (see *NeuronCore v2 Data Types*). Each data type provides benefits and drawbacks due to its dynamic range and numeric precision.

Type	Minimum	Maximum	Strength	Weakness
FP16	-65504	65504	Numeric Precision, High granularity, Mid-range numbers	Low range, medium precision
BF16	-3.40E+38	3.40E+38	Dynamic Range, Extremely small/large numbers	Low precision
TF32	-3.40E+38	3.40E+38	Dynamic Range, Extremely small/large numbers	Medium precision
FP32	-3.40E+38	3.40E+38	N/A	Larger model size, potentially slower computation

- FP16 provides a high density of representable values that are neither extremely small or extremely large. The density of representable values within the range is approximately an order of magnitude greater than BF16.
  - Conversion from FP32 to FP16 will perform well when values are relatively small but non-extreme (either very small or very large).
  - Conversion from FP32 to FP16 will perform badly if the original FP32 values are outside of the range of FP16. This will produce inf/-inf values and may result in NaN depending on the operation.
- BF16 provides a wider range of representable values which includes both very small and very large values. However, the overall density of representable values is usually lower than FP16 for more non-extreme values.

The range is nearly identical to the range of FP32 but because the number of bits is halved, this means the individual values are sparse.

- Conversion from FP32 to BF16 will perform well when the values are well-distributed throughout the range. Since BF16 covers the entire FP32 range, this means each original value can map to a relatively close downcast value.
- Conversion from FP32 to BF16 will perform badly when fine granularity is needed. Since BF16 granularity is sacrificed for greater range it will almost always map worse to values that are within the FP16 range.

12.2.5 Should I downcast operations to smaller Data Types?

This choice here is driven entirely by accuracy vs performance tradeoff. Casting operations to smaller 16-bit data types will provide a significant performance benefit but may end up sacrificing accuracy.

The compiler uses BF16 casting **by default** for matrix multiplication operations. The speedup from casting operations gives a significant performance boost and the range of representable values in BF16 allows for more safety compared to FP16 when the possible numeric range of input values is unknown.

The Neuron Compiler's `--auto-cast` and `--auto-cast-type` CLI options are used to direct the compiler to perform alternate casting operations. See the detailed list of the options in *Neuron v2 Compiler CLI Reference Guide*.

It is recommended that you start with compiling the model to achieve high performance (default), you can then test the accuracy of the application and, if needed, try the next higher precision casting option until the desired accuracy and performance are achieved.

The option combinations to consider in a typical flow are:

Note that compiler has to preserve the input/output (i/o) tensor types requested by Framework, therefore no casting is done on the i/o tensors. Additional speedup can be obtained by casting them in the Framework prior to compilation.

To learn how to configure the compiler options from within your application's framework, please see:

- *Developer Guide for Training with PyTorch Neuron*

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

12.3 Neuron Compiler FAQ (`neuronx-cc`)

Table of contents

- *Where can I compile to Neuron?*
- *What is the difference between* `neuron-cc` *and* `neuronx-cc`*?*
- *Should I use* `neuron-cc` *or* `neuronx-cc`*?*
- *My current neural network is based on FP32, how can I use it with Neuron?*
- *Which operators does Neuron support?*
- *Any operators that Neuron Compiler doesn't support?*
- *Will I need to recompile again if I updated runtime/driver version?*
- *I have a NEFF binary, how can I tell which compiler version generated it?*

- *How long does it take to compile?*
- *Why is my model producing different results compared to CPU/GPU?*
- *Do you support model* <insert model type>*?*

12.3.1 Where can I compile to Neuron?

The one-time compilation step from the standard framework-level model to NEFF binary may be performed on any EC2 instance or even on-premises.

We recommend using a high-performance compute server of choice (C5 or z1d instance types), for the fastest compile times and ease of use with a prebuilt DLAMI. Developers can also install Neuron in their own environments; this approach may work well for example when building a large fleet for inference, allowing the model creation, training and compilation to be done in the training fleet, with the NEFF files being distributed by a configuration management application to the inference fleet.

12.3.2 What is the difference between `neuron-cc` and `neuronx-cc`?

- `neuron-cc` is the Neuron Compiler with TVM front-end, ``neuron-cc``supports only *NeuronCore-v1 Architecture*.
- `neuronx-cc` is the Neuron Compiler with XLA fron-end, `neuronx-cc` currently supports *NeuronCore-v2 Architecture*, `neuronx-cc` support of *NeuronCore-v1 Architecture* is currently a *Roadmap Item*.

12.3.3 Should I use `neuron-cc` or `neuronx-cc`?

See *What is the difference between neuron-cc and neuronx-cc?*

12.3.4 My current neural network is based on FP32, how can I use it with Neuron?

Developers who want to train their models in FP32 for best accuracy can compile and deploy them with Neuron. The Neuron compiler automatically converts FP32 to internally supported datatypes, such as FP16 or BF16. You can find more details about FP32 data type support and performance and accuracy tuning in *Mixed Precision and Performance-accuracy Tuning (neuronx-cc)* or *Mixed precision and performance-accuracy tuning (neuron-cc)*. The Neuron compiler preserves the application interface - FP32 inputs and outputs. Transferring such large tensors may become a bottleneck for your application. Therefore, you can improve execution time by casting the inputs and outputs to FP16 or BF16 in the ML framework prior to compilation.

12.3.5 Which operators does Neuron support?

You can use the `neuronx-cc list-operators` command on the cli to list the operators. See *Neuron Compiler CLI Reference Guide (neuronx-cc)*.

To request support for new operators, open an issue on our GitHub forum.

12.3.6 Any operators that Neuron Compiler doesn't support?

Models with control-flow and dynamic shapes are not supported now. You will need to partition the model using the framework prior to compilation.

Note: Starting with *NeuronCore-v2 Architecture* Neuron supports control-flow and dynamic shapes.

Stay tuned and follow the *Neuron Roadmap*.

12.3.7 Will I need to recompile again if I updated runtime/driver version?

The compiler and runtime are committed to maintaining compatibility for major version releases with each other. The versioning is defined as major.minor, with compatibility for all versions with the same major number. If the versions mismatch, an error notification is logged and the load will fail. This will then require the model to be recompiled.

12.3.8 I have a NEFF binary, how can I tell which compiler version generated it?

** We will bring a utility out to help with this soon.

12.3.9 How long does it take to compile?

It depends on the model and its size and complexity, but this generally takes a few minutes.

12.3.10 Why is my model producing different results compared to CPU/GPU?

NeuronCore-v2 Architecture supports multiple casting modes for floating point numbers, each with associated implications for performance and accuracy. The default casting mode is a pragmatic balance between performance and accuracy, however on some models it may result in loss of precision.

See the `--auto-cast` and `--auto-cast-type` options in *Neuron Compiler CLI Reference Guide (neuronx-cc)* for details on how to adjust the casting mode.

12.3.11 Do you support model <*insert model type*>?

`neuronx-cc` has explicit support for select model families using the `--model-type` option, though many other model types are supported. You can also inspect supported operators using the `list-operators` sub-command. See th *Neuron Compiler CLI Reference Guide (neuronx-cc)* for details. More generally, support for new operators and models is continually being added. See our *Roadmap* for details.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

12.4 Neuron Compiler (neuronx-cc) release notes

Table of Contents

- *Neuron Compiler [2.3.0.4]*
- *Neuron Compiler [2.2.0.73]*
- *Neuron Compiler [2.1.0.76]*
  - *Known issues*
- *Supported Operators*

12.4.1 Neuron Compiler [2.3.0.4]

Date: 12/09/2022

- Added support for the following HLO operators: `rev (reverse)`.
- The `pow()` function can now handle both integer and floating-point exponents.
- Optimization enhancements and bug fixes to improve model execution performance.

12.4.2 Neuron Compiler [2.2.0.73]

Date: 10/27/2022

- Adding support for the following HLO operators: `LogicalNot`, `atan2` and `DynamicUpdateSlice` (for constant index).

12.4.3 Neuron Compiler [2.1.0.76]

Date: 10/5/2022

The Neuron Compiler is an Ahead-of-Time compiler that accelerates models for execution on NeuronCores. This release supports compiling models for training on a Trn1 instance using Pytorch Neuron. Users typically access the compiler via the Framework to perform model compilation, although it can also be run as a command line tool (*neuronx-cc*).

The Neuron Compiler supports compiling models for mixed precision calculations. The trn1 hardware supports matrix multiplication using FP16, BF16, and FP32 on its Matrix Multiplication Engine, and accumulations using FP32. Operators such as activations or vector operations are supported using FP16, BF16, and FP32. Tensor transpose can be accomplished in FP16, BF16, FP32, or TF32 datatypes. By default, scalar and vector operations on FP32 values will be done in FP32, while matrix multiplications are cast to BF16 and transpose operations are cast to FP32. This default casting will generate the highest performance for a FP32 trained model.

By default, the compiler will target maximum performance by automatically casting the model to mixed precision. It also provides an option (`--auto-cast`) that allows the user to make tradeoffs between higher performance and optimal accuracy. The decision on what option argument to use with the `--auto-cast` option will be application specific. Compiler CLI options can be passed to the compiler via the framework.

Known issues

- The Random Number Generator operation can be passed an initial seed value, however setting the seed is not supported in this release.
- The exponent value of the pow() function must be a compile-time integer constant.
- The compiler treats INT64 datatypes as INT32 by truncating the high-order bits. If possible, cast these values to 32 bits .
- Model compilation time is proportional to the model size and operators used. For some larger NLP models it may be upwards of 30 minutes.

12.4.4 Supported Operators

The following XLA operators are supported by the Neuron Compiler. Future releases will broaden model support by providing additional XLA operators defined in https://www.tensorflow.org/xla/operation_semantics.

The list of supported operators can also be retrieved from the command line using *neuronx-cc list-operators*.

Supported XLA Operators	Notes
Abs	
Add	
Allgather	
Allreduce	
Atan2	
Batchnorm	
Batchnormgrad	
Batchnorminference	
Broadcast	
BroadcastInDim	
Ceil	
Clamp	
Compare	
Concatenate	
Constant	
ConstantLiteral	
ConvertElementType	
Cos	
Customcall	
Div	
Dot	
DotGeneral	
DynamicUpdateSlice	Supports only for constant index
Eq	
Exp	
Floor	
Gather	Supports only disjoint start_index_map and remapped_offset_dims
Ge	
GetTupleElement	
Gt	
Iota	
Le	

continues on next page

Table 1 – continued from previous page

Supported XLA Operators	Notes
Log	
LogicalAnd	
LogicalNot	
Lt	
Max	
Min	
Mul	
Ne	
Neg	
Pad	
Pow	Exponent argument must be a compile-time integer constant
Reduce	Min, Max, Add and Mul are the only supported computations. Init_values must be constant
Reshape	
Rev (reverse)	
RngBitGenerator	Ignores user seed
RngUniform	
Rsqrt	
Scatter	
Select	
ShiftRightLogical	
Sign	
Sin	
Slice	
Sqrt	
Sub	
Tanh	
Transpose	
Tuple	

This document is relevant for: `Trn1`

Neuron Compiler for Inf1

API Reference Guide

This document is relevant for: `Inf1`

12.5 Neuron compiler CLI Reference Guide (`neuron-cc`)

This document describes the command line interface of the Neuron compiler. This reference is not relevant for applications that run neuron-cc from within a machine learning framework (TensorFlow-Neuron for example) since these options are passed from the framework directly to neuron-cc.

Using neuron-cc on the command line may be desirable for applications that do not use a framework, or customize existing frameworks. It is also possible to supply CLI commands to the framework as options to be passed through to the compiler.

12.5.1 Usage

Optional parameters are shown in square brackets. See the individual framework guides for the correct syntax.

Neuron Compiler CLI

neuron-cc `[options] <command> [parameters]`

Common options for the Neuron CLI:

- `--verbose` (string) default="WARN":

  Valid values:

  - `DEBUG`
  - `INFO`
  - `WARN`
  - `ERROR`

Use `neuron-cc <command> --help` for information on a specific command.

Available Commands:

- `compile`
- `list-operators`

neuron-cc `compile [parameters]`

Compile a model for use on the AWS Inferentia Machine Learning Accelerator.

```
neuron-cc compile <file names> --framework <value> --io-config <value> [--
↪neuroncore-pipeline-cores <value>] [--enable-fast-loading-neuron-binaries] [--
↪enable-fast-context-switch] [--fp32-cast cast-method] [--fast-math cast-method] [-
↪-output <value>]
```

Compile Parameters:

- `<file names>`: Input containing model specification. The number of arguments required varies between frameworks:

  - **TENSORFLOW**: A local filename or URI of a TensorFlow Frozen GraphDef (.pb); or the name of a local directory containing a TensorFlow SavedModel.

    See https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/framework/graph.proto for the associated .proto schema for TensorFlow Frozen GraphDefs. See https://www.tensorflow.org/guide/saved_model for more information on the SavedModel format.

  - **MXNET**: List of local filenames or URIs where input architecture .json file and parameter .param file are stored. These contains information related to the architecture of your graph and associated parameters, respectively.

- `--framework` (string): Framework in which the model was trained.

  Valid values:

  - `TENSORFLOW`
  - `MXNET`

  - XLA

- `--neuroncore-pipeline-cores` (int) (default=1): Number of neuron cores to be used in "NeuronCore Pipeline" mode. This is different from data parallel deployment (same model on multiple neuron cores). Refer to Runtime/Framework documentation for data parallel deployment options.

  Compile for the given number of neuron cores so as to leverage NeuronCore Pipeline mode.

  Note: This is not used to define the number of Neuron Cores to be used in a data parallel deployment (ie the same model on multiple Neuron Cores). That is a runtime/framework configuration choice.

- `--output` (string) (default="out.neff"): Filename where compilation output (NEFF archive) will be recorded.

- `--io-config` (string): Configuration containing the names and shapes of input and output tensors.

  The io-config can be specified as a local filename, a URI, or a string containing the io-config itself.

  The io-config must be formatted as a JSON object with two members "inputs" and "outputs". "inputs" is an object mapping input tensor names to an array of shape and data type. "outputs" is an array of output tensor names. Consider the following example:

```
{
 "inputs": {
    "input0:0": [[1,100,100,3], "float16"],
    "input1:0": [[1,100,100,3], "float16"]
 },
 "outputs": ["output:0"]
}
```

- `--enable-fast-loading-neuron-binaries` : Write the compilation output (NEFF archive) in uncompressed format which results in faster loading of the archive during inference.

- `--enable-fast-context-switch` : Optimize for faster model switching rather than inference latency. This results in overall faster system performance when your application switches between models frequently on the same neuron core (or set of cores). The optimization triggered by this option for example defers loading some weight constants until the start of inference.

- `--fast-math` : Controls tradeoff between performance and accuracy for fp32 operators. See more suggestions on how to use this option with the below arguments in *Mixed precision and performance-accuracy tuning (neuron-cc)*.

  - `all` (Default): enables all optimizations that improve performance. This option can potentially lower precision/accuracy.

  - `none` : Disables all optimizations that improve performance. This option will provide best precision/accuracy.

  - Tensor transpose options

    * `fast-relayout`: Only enables fast relayout optimization to improve performance by using the matrix multiplier for tensor transpose. The data type used for the transpose is either FP16 or BF16, which is controlled by the `fp32-cast-xxx` keyword.

    * `no-fast-relayout`: Disables fast relayout optimization which ensures that tensor transpose is bit-accurate (lossless) but slightly slower.

  - Casting options

  * `fp32-cast-all` (Default): Cast all FP32 operators to BF16 to achieve highest performance and preserve dynamic range. Same as setting `--fp32-cast all`.
  * `fp32-cast-all-fp16`: Cast all FP32 operators to FP16 to achieve speed up and increase precision versus BF16. Same setting as `--fp32-cast all-fp16`.
  * `fp32-cast-matmult`: Only cast FP32 operators that use Neuron Matmult engine to BF16 while using FP16 for matmult-based transpose to get better accuracy. Same as setting `--fp32-cast matmult`.
  * `fp32-cast-matmult-bf16`: Cast only FP32 operators that use Neuron Matmult engine (including matmult-based transpose) to BF16 to preserve dynamic range. Same as setting `--fp32-cast matmult-bf16`.
  * `fp32-cast-matmult-fp16`: Cast only FP32 operators that use Neuron Matmult engine (including matmult-based transpose) to fp16 to better preserve precision. Same as setting `--fp32-cast matmult-fp16`.

Important:

- `all` and `none` are mutually exclusive
- `all` is equivalent to using `fp32-cast-all fast-relayout` (best performance)
- `none` is equivalent to using `fp32-cast-matmult-bf16 no-fast-relayout` (best accuracy)
- `fp32-cast-*` options are mutually exclusive
- `fast-relayout` and `no-fast-relayout` are mutually exclusive
- The `fp32-cast-*` and `*-fast-relayout` options will overwrite the default behavior in `all` and `none`.
- For backward compatibility, the `--fp32-cast` option has higher priority over `--fast-math`. It will overwrite the FP32 casting options in any of the `--fast-math` options if `--fp32-cast` option is present explicitly.

- `--fp32-cast` : Refine the automatic casting of fp32 tensors. This is being replaced by a newer –fast-math.

Important:

- `--fp32-cast` option is being deprecated and `--fast-math` will replace it in future releases.
- `--fast-math` is introducing the `no-fast-relayout` option to enable lossless transpose operation.

The `--fp32-cast` is an interface for controlling the performance and accuracy tradeoffs. Many of the `--fast-math` values invoke (override) it.

- `all` (default): Cast all FP32 operators to BF16 to achieve speed up and preserve dynamic range.
- `matmult`: Cast only FP32 operators that use Neuron Matmult engine to BF16 while using fp16 for matmult-based transpose to get better accuracy.
- `matmult-fp16`: Cast only FP32 operators that use Neuron Matmult engine (including matmult-based transpose) to fp16 to better preserve precision.

- `matmult-bf16`: Cast only FP32 operators that use Neuron Matmult engine (including matmult-based transpose) to BF16 to preserve dynamic range.
- `all-fp16`: Cast all FP32 operators to FP16 to achieve speed up and better preserve precision.

Log Levels:

Logs at levels "trace", "debug", and "info" will be written to STDOUT.

Logs at levels "warn", "error", and "fatal" will be written to STDERR.

Exit Status

0 - Compilation succeeded

>0 - An error occurred during compilation.

Examples

Compiling a saved TensorFlow model:

```
neuron-cc compile test_graph_tfmatmul.pb --framework TENSORFLOW --io-config␣
↪test_graph_tfmatmul.config
```

Compiling a MXNet model:

```
neuron-cc compile lenet-symbol.json lenet-0001.params --framework MXNET --
↪neuroncore-pipeline-cores 2 --output file.neff
```

Compiling an XLA HLO:

```
neuron-cc compile bert-model.hlo --framework XLA  --output file.neff
```

neuron-cc `list-operators [parameters]`

Returns a newline ('n') separated list of operators supported by the NeuronCore.

- **TENSORFLOW**: Operators will be formatted according to the value passed to the associated REGISTER_OP("OperatorName") macro.

  See https://www.tensorflow.org/guide/create_op#define_the_op_interface for more information regarding operator registration in TensorFlow.

- **MXNET**: Operator names will be formatted according to the value passed to the associated NNVM_REGISTER_OP(operator_name) macro.
- **XLA**: Operator names will be formatted according to the value used by XLA compiler in XlaBuilder.

  See https://www.tensorflow.org/xla/operation_semantics for more information regarding XLA operator semantics in XLA interface.

```
neuron-cc list-operators --framework <value>
```

- `--framework` (string): Framework in which the operators were registered.

  Valid values:

  - `TENSORFLOW`
  - `MXNET`
  - `XLA`

Exit Status

0 - Call succeeded

>0 - An error occurred

Example

```
$ neuron-cc list-operators --framework TENSORFLOW
AddN
AdjustContrastv2
CheckNumbers
...
```

This document is relevant for: `Inf1`

Developer Guide

This document is relevant for: `Inf1`

12.6 Mixed precision and performance-accuracy tuning (`neuron-cc`)

Table of contents

- *Neuron Hardware*
- *Performance-accuracy tradeoffs for models trained in FP32*
- *Compiler casting options*
  - *`--fast-math` option*

The Neuron Compiler supports machine learning models with FP32, FP16 and BF16 (Bfloat16) tensors and operators. The Neuron hardware supports a mix of 32 and 16 bit datatypes. The available auto-cast methods and their performance / accuracy trade-offs are explained in this document.

12.6.1 Neuron Hardware

The Neuron hardware supports matrix multiplication using FP16 or BF16 on its Matmult Engine, and accumulations using FP32. Similarly, operators such as activations or vector operations are supported using FP16, BF16 and FP32. Neuron supports tensor transpose in two ways - by fast matrix multiplication in FP16/BF16 or by slower byte-by-byte data movements.

12.6.2 Performance-accuracy tradeoffs for models trained in FP32

Models that are trained using FP32 data types can be deployed on Neuron through ahead of time compilation using the *Neuron Compiler*.

By default, the Neuron Compiler will **cast all FP32 tensors, weights and operations to BF16**. Only partial sums are left in FP32. The default, casting will generate the highest performance for a FP32 trained model.

Using the `--fast-math` CLI option, you can choose the right tradeoff between performance and accuracy. The tradeoff usually is between achieving high performance or optimal accuracy, and decision what settings to use will be application specific.

It is recommended that the you start with compiling the model to achieve the high performance (default), you can then test the accuracy of the application and, if needed, try the next higher precision casting option until the desired accuracy and performance are achieved. A typical flow can be:

1. You can compile without options (default) or with `--fast-math all` which will optimize for performance.
2. If accuracy is not sufficient you can try `--fast-math fp32-cast-matmult`
3. If accuracy is not sufficient you can try `--fast-math fp32-cast-matmult no-fast-relayout`
4. If accuracy is not sufficient you can try `--fast-math none` which will optimize for accuracy .

Between step 2 and step 3, and between step 3 and step 4 you have additional options that can provide different level of accuracy and which are explained in the below section.

Note that compiler has to preserve the input/output (i/o) tensor types requested by Framework, therefore no casting is done on the i/o tensors. Additional speedup can be obtained by casting them in the Framework prior compilation.

To learn how to use compiler command line interface (CLI) options with your application's framework, please see *PyTorch-Neuron trace python API*, *TensorFlow 1.x (tensorflow-neuron) Compilation API* and *TensorFlow 2.x (tensorflow-neuron) Tracing API*.

12.6.3 Compiler casting options

`--fast-math` option

The `--fast-math` option is intended to replace the `--fp32-cast` option. It is recommended to to start using or migrating to `--fast-math` option. The `--fast-math` option provides the same level of functionality as the `--fp32-cast` option in addition to the following:

- The `--fast-math` option introduces the `no-fast-relayout` option to enable lossless transpose operation. This was not possible with the `--fp32-cast` option.
- The `--fast-math` option provides finer control than the `--fp32-cast` option. The transpose operation and the cast operation are controlled independently:
  - `no-fast-relayout` and `fast-relayout` provide control for the transpose operation.
  - `fp32-cast-*` provide control for casting.

See the detailed list of the options in *Neuron compiler CLI Reference Guide (neuron-cc)*.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

12.7 Neuron Compiler FAQ (neuron-cc)

Table of contents

- *Where can I compile to Neuron?*
- *My current Neural Network is based on FP32, how can I use it with Neuron?*
- *What are some of the important compiler defaults I should be aware of?*
- *Which operators does Neuron support?*
- *Any operators that Neuron doesn't support?*
- *Will I need to recompile again if I updated runtime/driver version?*
- *I have a NEFF binary, how can I tell which compiler version*
- *How long does it take to compile?*

12.7.1 Where can I compile to Neuron?

The one-time compilation step from the standard framework-level model to NEFF binary may be performed on any EC2 instance or even on-premises.

We recommend using a high-performance compute server of choice (C5 or z1d instance types), for the fastest compile times and ease of use with a prebuilt DLAMI. Developers can also install Neuron in their own environments; this approach may work well for example when building a large fleet for inference, allowing the model creation, training and compilation to be done in the training fleet, with the NEFF files being distributed by a configuration management application to the inference fleet.

12.7.2 My current Neural Network is based on FP32, how can I use it with Neuron?

Developers who want to train their models in FP32 for best accuracy can compile and deploy them with Neuron. The Neuron compiler automatically converts FP32 to internally supported datatypes, such as FP16 or BF16. You can find more details about FP32 data type support and performance and accuracy tuning in *Mixed precision and performance-accuracy tuning (neuron-cc)*. The Neuron compiler preserves the application interface - FP32 inputs and outputs. Transferring such large tensors may become a bottleneck for your application. Therefore, you can improve execution time by casting the inputs and outputs to FP16 or BF16 in the ML framework prior to compilation for Inferentia.

12.7.3 What are some of the important compiler defaults I should be aware of?

The compiler compiles the input graph for a single NeuronCore by default. Using the `--neuroncore-pipeline-cores` option directs the compiler to partition so as to run on a specified number of NeuronCores. This number can be less than the total available NeuronCores on an instance. See *Inferentia Architecture* for more information on NeuronCores.

12.7.4 Which operators does Neuron support?

see *Neuron Supported operators*.

You can also use the "neuron-cc list-operators" command on the cli to list the operators. See neuron-cc-list-operators

If your model contains operators missing from the above list, and you can't reach your performance goals, please post a message on the Neuron developer forum or open a github issue to let us know.

12.7.5 Any operators that Neuron doesn't support?

Models with control-flow and dynamic shapes are not supported. You will need to partition the model using the framework prior to compilation. See the neuron-cc.

12.7.6 Will I need to recompile again if I updated runtime/driver version?

The compiler and runtime are committed to maintaining compatibility for major version releases with each other. The versioning is defined as major.minor, with compatibility for all versions with the same major number. If the versions mismatch, an error notification is logged and the load will fail. This will then require the model to be recompiled.

12.7.7 I have a NEFF binary, how can I tell which compiler version

generated it?** We will bring a utility out to help with this soon.

12.7.8 How long does it take to compile?

It depends on the model and its size and complexity, but this generally takes a few minutes.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

12.8 Neuron Compiler (`neuron-cc`) for Inf1 Release Notes

Table of contents

- *Introduction*
- *Known issues and limitations - updated 11/23/2022*
- *Neuron Compiler release [1.13.3.0]*
- *Neuron Compiler release [1.11.7.0]*
- *Neuron Compiler release [1.11.4.0]*
- *Neuron Compiler release [1.10.3.0]*
- *Neuron Compiler release [1.9.1.0]*
- *Neuron Compiler release [1.8.5.0]*
- *Neuron Compiler release [1.8.2.0]*

- *Neuron Compiler release [1.7.3.0]*
- *[1.6.13.0]*
- *[1.5.5.0]*
- *[1.4.0.0]*
- *[1.3.0.0]*
- *[1.2.7.0]*
- *[1.2.2.0]*
- *[1.1.7.0]*
- *[1.0.24045.0]*
- *[1.0.20600.0]*
- *[1.0.18001.0]*
- *[1.0.17937.0]*
- *[1.0.16861.0]*
- *[1.0.15275.0]*
- *[1.0.12696.0]*
- *[1.0.9410.0]*
- *[1.0.7878.0]*
- *[1.0.6801.0]*
- *[1.0.5939.0]*
- *[1.0.5301.0]*
- *[1.0.4680.0]*

12.8.1 Introduction

This document lists the release notes for AWS Neuron compiler. The Neuron Compiler is an ahead-of-time compiler that ensures Neuron will optimally utilize the Inferentia chips.

Operator-support for each input format is provided directly from the compiler.

```
neuron-cc list-operators --framework {TENSORFLOW | MXNET | XLA}
```

The supported operators are also listed here:

Tensorflow: *TensorFlow Neuron (tensorflow-neuron (TF1.x)) Supported operators*

Pytorch: *PyTorch Neuron (torch-neuron) Supported operators*

XLA: neuron-cc-ops-xla

Apache MXNet (Incubating): *Neuron Apache MXNet (Incubating) Supported operators*

12.8.2 Known issues and limitations - updated 11/23/2022

- There is a known issue of increased latency and lower throughput when MLM head is compiled along with BERT model. The workaround is to compile them separately and feed the raw Bert into the head.
- *TensorFlow 2.x* - In this release supported operators are limited to BERT-like models, specifically no conv2d or reduce-window operators are available.
- *Control flow* Neuron only supports control flow operators which are static at compile time. For example static length RNN, top-k, sort.
- *Data layout* The Neuron compiler supports multiple data layout format (NCHW, NHWC, ...). Non-CNHW input/output data-layouts will require Neuron to insert additional transpose operations, causing a degradation in performance.
- *Primary inputs in NeuronCore Pipeline mode* When a neural network is executed in NeuronCore Pipeline mode, only the first operator in a neural network can receive primary inputs from the host.
- *Reduce data type* INT8 data type is not currently supported by the Neuron compiler.
- *NeuronCore Pipeline:* NeuronCorePipeline mode provides low-latency and high-throughput for small batch sizes. We recommend to start testing with batch=1 and gradually increase batch size to fine tune your model throughput and latency performance.
- *Large input tensors* support varies by model. On some models the large input tensors (eg 1024x1024) may result in lower performance or exceeding hardware or compile-time limits, especially on models where the large input tensor is used by many downstream operators. Workarounds may include use of smaller batch, see *Neuron Batching*
- *Conv2d operator* is mapped to Inferentia except for specific cases of extremely large tensors and specific parameters.
- *Conv3d operator* performance is limited when the operator has small number of input channels (< 64).
- FP64 and INT64 input and output tensors are not supported. Please cast to FP32/INT32 in the machine learning framework, prior compiling for Neuron.

12.8.3 Neuron Compiler release [1.13.3.0]

Date: 11/23/2022

- Resolved long compile-times when compiling the YOLOv5 and YOLOv6 models. [GitHub · aws-neuron-sdk · #434]
- Improved the layout algorithm to resolve an issue compiling a transformer-based text recognition model. [GitHub · aws-neuron-sdk · #410]
- Support was added for additional XLA operators

12.8.4 Neuron Compiler release [1.11.7.0]

Date: 08/02/2022

- Fixed a bug for correct handling of mxnet dropout instruction when mode is set as 'training' while performing inference.

12.8.5 Neuron Compiler release [1.11.4.0]

Date: 04/29/2022

- Solved an issue that caused a "false positive" reporting of a data race that may occur due to address overlap.
- Minor bug fixes.

12.8.6 Neuron Compiler release [1.10.3.0]

Date: 03/25/2022

- Minor bug fixes.

12.8.7 Neuron Compiler release [1.9.1.0]

Date: 01/20/2022

- Fixed an issue with frontend compiler for fused operators that was reported in github #362.

12.8.8 Neuron Compiler release [1.8.5.0]

Date: 01/05/2022

New in this release

- Minor bug fixes.

12.8.9 Neuron Compiler release [1.8.2.0]

Date: 12/15/2021

New in this release

- Performance enhancements as a result of improved layout and DMA optimizations.
- Minor bug fixes.

12.8.10 Neuron Compiler release [1.7.3.0]

Date: 10/27/2021

New in this release

- The compiler's list-operators command can now display the supported TensorFlow 2.x operators.
- Support added for new operators in TensorFlow 1.x - ArgMax and ArgMin.
- Introducing the `--fast-math` option for better fine-tuning of accuracy/performance. See *Mixed precision and performance-accuracy tuning (neuron-cc)*

12.8.11 [1.6.13.0]

Date 08/12/2021

New in this release

- TensorFlow 2.x - First support of TensorFlow 2.x. The support is limited to operators in BERT-like models and was tested with Huggingface BERT small, base, large and DistillBert.

Resolved issues

- Fixed compiler backend issue in Tensor_tensor argument distance, github #269

12.8.12 [1.5.5.0]

Date 07/02/2021

Summary

- Robustness and performance improvements.

New in this release

- Added –enable-fast-context-switch to optimize for faster model switching rather than inference latency.
- Deprecated support for ONNX
- Improved robustness of Conv3d
- Corrected compilation error "too many instructions" in DLRM model

12.8.13 [1.4.0.0]

Date 5/28/2021

Summary

- Performance improvements, and usability improvements.

New in this release

- Added uncompressed NEFF format for faster loading models prior inference. Enable it by –enable-fast-loading-neuron-binaries. Some cases of large models may be detrminentally impacted as it will not be compressed but many cases will benefit.
- Corrected compilation error in specific arguments of ResizeBilinear operator

12.8.14 [1.3.0.0]

Date 4/30/2021

Summary

- Performance improvements, new operators, and usability improvements.

New in this release

- Improved performance of batched CNN models like resnet50 with the default compiler options by 10%.
- Improved performance of bert base sequence 128 batch 6 by upto 16%
- Added support for group and depth wise convolution (with limited performance when the number of input channels is small).
- Added more detailed debug names to support for tensorboard.

Resolved Issues

- Corrected potential race condition in overwriting tiles of output tensors.
- Fixed various issues in pipelined inference by enabling fine grain partitioning by default.

12.8.15 [1.2.7.0]

Date 2/24/2021

Summary

Fix for CVE-2021-3177.

12.8.16 [1.2.2.0]

Date 1/30/2021

Summary

Added suport for multiple new operators (see operators list) for Tensoflow and MXNET. Improved inference performance of language, object recognition models on single as well as multiple pipelined cores using neuroncore-pipeline.

New in this release

- The following models are now supported: Resnext 224x224, specific BERT variations applied to natural language processing and translation.
- A number of new operators is now supported on Inferentia, see the full lists *TensorFlow Neuron (tensorflow-neuron (TF1.x)) Supported operators*

  and *Neuron Apache MXNet (Incubating) Supported operators*
- Improved inference performance on yolov4 BERT base sequence 64 (on 16 pipelined cores) and openpose 184.

Resolved Issues

- Corrected a random failure to compile Resnet50 batch 5
- Corrected numerical inaccuracy in RSQRT and related operators for tensors with very large values (> 1e20)

12.8.17 [1.1.7.0]

Date 12/23/2020

Summary

Added suport for PyTorch Yolo V4, a new Framework-visible progress bar and improved inference performance. We continue to streamline the compiler usability by removing the need for options passed to control behavior. We are aiming to remove the need for such options entirely. Some tutorials have been updated to reflect this, but Resnet50 remains in need of these options to achieve maximum performance. Other useability improvements have been added, such as the compiler progress bar. As always, please let us know if there are other areas that we can improve.

New in this release

- Pytorch Yolo V4 is now supported.
- Added a compiler progress bar when compilation is invoked from the Framework. This allows the user to see that progress continues as compilation proceeds, which is useful when compilation takes several minutes. A dot is printed every 20 seconds.
- Improved inference performance of Tensorflow BERT base seq 256 batch 3 by 10% .

Resolved Issues

- Resolved issue with depthwise convolution that manifests as a type check error

12.8.18 [1.0.24045.0]

Date 11/17/2020

Summary

Improved performance for pipelined execution (NeuronCore Pipeline).

New in this release

- NeuronCore Pipeline: improved partitioning to enable better static weights loading to cache.

Resolved Issues

- –static-weights : No longer needed. As this is shown in some examples, please remove the option since the compiler now performs this auto-detection by default.
- –num-neuroncores renamed to –neuroncore-pipeline-cores. The prior option form is still functional (backwards compatible) and will be removed in future releases.
- –batching_en: Resolved compilation failure of ResNet50 FP32 batch 1 on Ubuntu16 when "–batching_en" was used.

12.8.19 [1.0.20600.0]

Date 9/22/2020

Summary

Various performance improvements - both compilation time and inference speed of object recognition models.

- Compiler optimization '-O2' option is now enabled by default.

New in this release

- Improved inference performance of YOLO v3, YOLO v4, VGG16, SSD300. BERT models were improved by an additional 10%.
- Modifed such that -O2 is now the default behavior and does not need to be specified. Note: some tutorials still explicitly specify "-O2". These will be modified in forthcoming updates.

Resolved Issues

- Sped up compilation of large models that were taking hours to sub-40 minute.

12.8.20 [1.0.18001.0]

Date 8/08/2020

Summary

Various performance improvements.

New in this release

Improved performance of BERT base with -O2

Resolved Issues

- n/a

12.8.21 [1.0.17937.0]

Date 8/05/2020

Summary

Various improvements.

12.8.22 [1.0.16861.0]

Date 7/16/2020

Summary

This release has some bug fixes and some functional and performance improvements to support compilation of several neural networks.

New in this release

This release

- Supports compilation of PoseNet, tested for images of specific resolutions upto 736.
- Update the -O2 with a new memory allocator to reduce spilling to DRAM
- Improved performance of the '-O2' on BERT base, and openpose pose network.

Resolved Issues

- Resolved compilation error in Vgg16 batch 1

Other Notes

- Some versions of Inception network may fail to compile in Tensorflow on Ubuntu 16 in conda environment. The symptom is neuron-cc backend data race error. As a workaround use Ubuntu 18, Amazon Linux 2, or virtual env, or use neuron-cc with flag -O2.

Warning: Starting with Neuron 1.14.0, Ubuntu 16 is no longer supported

12.8.23 [1.0.15275.0]

Date 6/11/2020

Summary

This release has some bug fixes and some functional and performance improvements to support compilation of several neural networks.

New in this release

This release

- Supports compilation of PoseNet for images of specific resolutions upto 400x400.
- Improves performance of resnet152.
- Supports a new command line option '-O2' that can help with handling of large tensor inputs for certain models.
- increase NEFF versions to 1.0. This means new NEFFs compiled from this release forward are not compatible with older versions of Neuron Runtime prior to May, 2020 (1.0.6905.0) release. Please update the Neuron Runtime when using NEFF version 1.0.

Resolved Issues

- Compilation issues on prosotron encoder, decoder neural networks.

Other Notes

Dependencies

- This version creates NEFF 1.0 thus may require update of neuron-rtd if older than May 2020 release.

dmlc_nnvm==1.0.2574.0 dmlc_topi==1.0.2574.0 dmlc_tvm==1.0.2574.0 inferentia_hwm==1.0.1362.0 islpy==2018.2

12.8.24 [1.0.12696.0]

Date 5/11/2020

Summary

Bug fixes and some functional and performance improvements to several neural networks.

New in this release

- This version supports compilation of unmodified Tensorflow BERT with batch size 1, 4, 6 for input sequence 128.
- Improved Tensorflow BERT batch 4 sequence 128 performance to 45% of the accelerator peak (from 34%).
- Support for MXNET BERT base batch 8 compilation
- Support for TF Resnet152 batch 2 compilation
- Most compiler messages are migrated from cout to logging mechanisms with verbosity control

Resolved Issues

- Fixed failure to compile unmodified Tensorflow BERT model for small batches
- Fixed run-to-run-variability in OneHot operator implementation
- Robustness improvements for ParallelWavenet and transformer decoder networks

Other Notes

Dependencies

```
dmlc_nnvm==1.0.2356.0
dmlc_topi==1.0.2356.0
dmlc_tvm==1.0.2356.0
inferentia_hwm==1.0.1294.0
islpy==2018.2
```

12.8.25 [1.0.9410.0]

Date 3/26/2020

Summary

Bug fixes and some functional and performance improvements to several neural networks.

New in this release

- Support compilation of modified SSD-300 (*Running SSD300 with AWS Neuron*)
- Improved inference performance in natural language processing networks (such as prosotron encoder) by 45%

Resolved Issues

- Eliminated redundant fp32 to bfloat16 cast on input and output tensors

Known issues and limitations

- See previous releases.

Other Notes

- Added support for faster iteration on recurrent networks (aka auto-loop)

Dependencies

```
dmlc_nnvm==1.0.2049.0
dmlc_topi==1.0.2049.0
pip install --upgrade dmlc_tvm==1.0.2049.0
inferentia_hwm==1.0.897.0
islpy==2018.2
```

12.8.26 [1.0.7878.0]

Date 2/27/2020

Summary

Bug fixes and minor performance improvements.

New in this release

None

Resolved Issues

- Corrected image resize operator functionallity
- Compiler internal enhancements made that will benefit models such as BERT

Known issues and limitations

- See previous releases.

Other Notes

Dependencies

```
dmlc_nnvm-1.0.1826.0
dmlc_topi-1.0.1826.0
dmlc_tvm-1.0.1826.0
inferentia_hwm-1.0.897.0
islpy-2018.2
```

12.8.27 [1.0.6801.0]

Date 1/27/2020

Summary

Bug fixes and some performance enhancement related to data movement for BERT-type neural networks.

New in this release

None

Resolved Issues

- Improved throughput for operators processed in the Neuron Runtime CPU. As an example: execution of 4 single NeuronCore NEFF models of ResNet50 v2 float16 batch = 5 in parallel on an inf1.1xlarge sped up by 30%.
- Corrected shape handling in Gather(TensorFlow)/Take(MXNet) operators that are processed by the Neuron Runtime in the Neuron Runtime vCPU, which resolves a possible crash in Neuron Compiler when compiling models with these operators with some shapes.
- Added support for TensorFlow *OneHot* operator (as a Neuron Runtime CPU operator).
- Added more internal checking for compiler correctness with newly defined error messages for this case.

```
"Internal ERROR: Data race between Op1 'Name1(...) [...]' and Op2 'Name2(...) [...]'"
```

- Fixed out-of-memory issue introduced in 1.0.5939.0 such that some large models (BERT) compiled on instances with insufficient host memory would cause the runtime to crash with an invalid NEFF. This is actually a compiler error, but due to additional script layers wrapping this in the *Running TensorFlow BERT-Large with AWS Neuron*, this would have likely been seen as a runtime error like this:

```
2020-01-09 13:40:26.002594: E tensorflow/core/framework/op_segment.cc:54] Create kernel␣
↪failed: Invalid argument: neff is invalid
2020-01-09 13:40:26.002637: E tensorflow/core/common_runtime/executor.cc:642] Executor␣
↪failed to create kernel. Invalid argument: neff is invalid
[[{{node bert/NeuronOp}}]]
```

Known issues and limitations

See previous release notes. Some tutorials show use of specific compiler options and flags, these are needed to help provide guidance to the compiler to achieve best performance in specific cases. Please do not use in cases other than as shown in the specific tutorial as results may not be defined. These options should be considered experimental and will be removed over time.

Other Notes

Dependencies

```
dmlc_nnvm-1.0.1619.0
dmlc_topi-1.0.1619.0
dmlc_tvm-1.0.1619.0
inferentia_hwm-1.0.839.0
islpy-2018.2
```

12.8.28 [1.0.5939.0]

Date 12/20/2019

Summary

Bug fixes and some performance enhancement for NeuronCore Pipeline.

New in this release

Resolved Issues

- Fixed pipeline execution on more than 10 NeuronCores
- Improved NeuronCores Pipeline execution by improving data exchange efficiency between NeuronCores
- Added warning for unaligned memory access
- Fixed handling of cast on input FP32 tensor
- Improved handling of data layouts and transpose

- Improved dead-code elimination
- Improved efficiency of compute engine synchronization
- Improved efficiency of data transfers within the Neuron code

Known issues and limitations

See previous release notes. Some tutorials show use of specific compiler options and flags, these are needed to help provide guidance to the compiler to achieve best performance in specific cases. Please do not use in cases other than as shown in the specific tutorial as results may not be defined. These options should be considered experimental and will be removed over time.

Other Notes

Dependencies

- dmlc_nnvm-1.0.1416.0
- dmlc_topi-1.0.1416.0
- dmlc_tvm-1.0.1416.0
- inferentia_hwm-1.0.720.0
- islpy-2018.2

12.8.29 [1.0.5301.0]

Date 12/1/2019

Summary

New in this release

Resolved Issues

- Added warning for unsupported operators and convolution sizes
- Added warning for unsupported layout / upsampling
- Added support for Relu6, AddV2, BatchMatmulV2 operators
- Added support for default MXNet outputs in –io-config
- Improved performance of batched inference for convolutional networks
- Fixed MatMult column size 1
- Fixed bf16 constant loading
- Fixed Conv2D tile accumulation

Known Issues and Limitations

See previous release notes. Resolved issues are shown in Resolved Issues.

Other Notes

Please install g++ on AMIs without g++ pre-installed (i.e. server AMIs):

```
# Ubuntu
sudo apt-get install -y g++
```

```
# Amazon Linux
sudo yum nstall -y gcc-c++
```

Supported Python versions:

- 3.5, 3.6, 3.7

Supported Linux distributions:

- Ubuntu 16, Ubuntu 18, Amazon Linux 2

Dependencies

- dmlc_nnvm-1.0.1328.0
- dmlc_topi-1.0.1328.0
- dmlc_tvm-1.0.1328.0
- inferentia_hwm-1.0.674.0
- islpy-2018.2

12.8.30 [1.0.4680.0]

Date: 11/25/2019

New in this release

N/A, this is the first release.

Resolved issues

N/A, this is the first release.

Known issues and limitations

1. **Control flow** Inferentia has a limited support for control flow. In general, Neuron can only support control flow operators which are static at compile time, i.e. static length RNN, top-k, sort, …
2. **Size of neural network** The size of neural network is influenced by a) type of neural network (CNN, LSTM, MLP) , b) number of layers, c) sizes of input (dimension of the tensors, batch size, …). The current Neuron compiler release has a limitation in terms of the size of neural network it could effectively optimize. As a result, we limit CNN models (e.g. ResNet) to have an input size of up to 480x480 FP16, batch size of 4; LSTM models (e.g. GNMT) are limited to a time step limit of up to 900; MLP models (like BERT) are limited up to sequence-length equal 128, batch=8.
3. **Data layout** The Neuron compiler supports multiple data layout formats (NCHW, NHWC, …). Non-CNHW input/output data-layouts will require Neuron to insert additional *transpose* operations, causing a degradation in performance.
4. **Object detection models** Computer-vision object detection and segmentation models are not supported by the current release.
5. **Reduce data type** INT8 data type is not currently supported by the Neuron compiler.
6. **Tensor residency** When a sub-graph that is executed on the host is communicating with a sub-graph that is executing on Neuron cores, tensors are copied via the communication queues between the host and Inferentia memory for each inference, which may result in end-to-end performance degradation.
7. **Primary inputs in NeuronCore Pipeline mode** When a neural network is executed in NeuronCore Pipeline mode, only the first operator in a neural network can receive primary inputs from the host.

Other Notes

Dependencies

- nnvm: dmlc_nnvm-1.0.1219.0
- topi: dmlc_topi-1.0.1219.0
- tvm: dmlc_tvm-1.0.1219.0
- hwm: inferentia_hwm-1.0.602.0
- islpy: islpy-2018.2+aws2018.x.73.0

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

12.9 Neuron Supported operators

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

THIRTEEN

NEURON TOOLS

Neuron provides debugging and profiling tools with the visualization support of the TensorBoard plugin. The Neuron helper tools assist in best practices for model onboarding and performance optimizations. The debugging and profiling tools provide monitoring of runtime and performance metrics insights.

This document is relevant for: `Inf1`, `Trn1`

13.1 System Tools

This document is relevant for: `Inf1`, `Trn1`

13.1.1 Neuron Monitor User Guide

Table of contents

- *Overview*
- *Using neuron-monitor*
  - *Configuration file example*
  - *Neuron applications tagging*
  - *JSON objects and fields in the configuration file*
  - *Neuron Runtime-level metric groups*
  - *System-wide metric groups*
- *Execution model*
- *The JSON output format*
- *Neuron application level metric groups*
  - *neuroncore_counters*
  - *execution_stats*
  - *memory_used*
  - *neuron_runtime_vcpu_usage*
- *System level metric groups*
  - *neuron_hw_counters*

- *vcpu_usage*
- *memory_info*

- *Companion scripts*
  - *neuron-monitor-cloudwatch.py*
  - *neuron-monitor-prometheus.py*

Overview

neuron-monitor collects metrics and stats from the Neuron Applications running on the system and streams the collected data to `stdout` in `JSON` format. It is provided as part of the `aws-neuron-tools` package.

These metrics and stats are organized into **metric groups** which can be configured by providing a configuration file as described in *Using neuron-monitor*

When running, **neuron-monitor** will:

- Collect the data for the metric groups which, based on the elapsed time since their last update, need to be updated
- Take the newly collected data and consolidate it into a large report
- Serialize that report to JSON and stream it to stdout from where it can be consumed by other tools - such as the sample *neuron-monitor-cloudwatch.py* and *neuron-monitor-prometheus.py* scripts.
- Wait until at least one **metric group** needs to be collected and repeat this flow

Using neuron-monitor

neuron-monitor CLI

neuron-monitor `[parameters]`

neuron-monitor accepts the following optional parameters:

- `--verbose` (int) default=0: Can be 0 to 4, and controls the amount of debugging and verbose information sent to stderr; **0: no output**, **4: maximum verbosity**
- `-c, --config-file` (string): Allows specifying a valid path to a neuron-monitor JSON configuration file

Example:

```
neuron-monitor -c monitor.conf
```

Not specifying any configuration file will enable collecting all the metric groups with a period of 5 seconds for all currently running Neuron applications.

Configuration file example

Example of a configuration file which enables all available **metric groups** for every running Neuron application, with a global update period of 1 second and sets an update period of 2 seconds for the `"neuron_hw_counters"` metric group:

```
{
  "period": "1s",
  "neuron_runtimes": [
    {
      "tag_filter": ".*",
      "metrics": [
        {
          "type": "neuroncore_counters"
        },
        {
          "type": "memory_used"
        },
        {
          "type": "neuron_runtime_vcpu_usage"
        },
        {
          "type": "execution_stats"
        }
      ]
    }
  ],
  "system_metrics": [
    {
      "type": "vcpu_usage"
    },
    {
      "type": "memory_info"
    },
    {
       "period": "2s",
       "type": "neuron_hw_counters"
    }
  ]
}
```

Neuron applications tagging

In order to make application monitoring easier, Neuron applications can be tagged with a 255 character string which identifies that app. Tagging is done using the `NEURON_PROCESS_TAG` environment variable.

For example: `NEURON_PROCESS_TAG=my_app_1 python training.py` will associate the `my_app_1` tag with that Python application. If `NEURON_PROCESS_TAG` is not specified, the application's PID will be used as a TAG.

This tag will be used by neuron-monitor to filter Neuron applications.

JSON objects and fields in the configuration file

- `"neuron_runtimes"` - array of objects specifying which Neuron Applications to monitor and what metric groups are enabled for each of them
  - `"tag_filter"` - a regex which will be used to filter Neuron applications tags in order to determine if they will be monitored (optional)
  - `"metrics"` - array of objects specifying which metric groups to capture for this Neuron application
    - `"type"` - type of metric group
- `"period"` - this field applies to **metric group** objects and sets the amount of time between two updates for that metric group
  - if can be specified as part of the **root** and/or **neuron_runtime** objects where it applies to all their children, and/or as part of a **metric group** object
  - if there's no period specified, a default value of **5 seconds** will be used
- `"system_metrics"` - array of objects specifying which system level metric groups are enabled

Neuron Runtime-level metric groups

- *neuroncore_counters* - NeuronCore related metrics
- *memory_used* - data on the amount of memory used by the Neuron application
- *vcpu_usage* - Neuron application vCPU utilization data
- *execution_stats* - Neuron application execution stats, including error count and latency

System-wide metric groups

- *vcpu_usage* - system-wide vCPU usage
- *memory_info* - system-wide memory usage
- *neuron_hw_counters* - counters for correctable and uncorrectable memory ecc events

Execution model

neuron-monitor waits for one or more **metric groups** to be up for update, then collects the corresponding data, consolidates it into a report which is streamed to stdout as a JSON and goes back to waiting.

The JSON output format

Whenever the report gets updated, a complete JSON is written to stdout. This is its structure:

```
{
  "neuron_runtime_data": [
    {
      "pid": 0,
      "address": "",
      "neuron_runtime_tag", "my_app_1",
      "error": "",
      "report": {
        "neuroncore_counters": {
            [...]
        },
        "execution_stats": {
            [...]
        },
        "memory_used": {
            [...]
        },
        "neuron_runtime_vcpu_usage": {
            [...]
        }
      }
```

(continues on next page)

(continued from previous page)

```
    }
  ],
  "system_data": {
    "neuron_hw_counters": {
            [...]
    },
    "vcpu_usage": {
            [...]
    },
    "memory_info": {
            [...]
    }
  },
  "instance_info": {
            [...]
  },
  "neuron_hardware_info": {
            [...]
  }
}
```

- "neuron_runtime_data" is an array containing one entry per each Neuron application which passes the filter specified in the settings file
  - "pid" is the pid of this Neuron application
  - "neuron_runtime_tag" is the configured tag for the Neuron application
  - "error" specifies any error that occurred when collecting data from this Neuron application
  - "report" will contain the results for the Neuron application-level metric groups; their formats are described below
- "system_data" has a similar structure to "neuron_runtime_data"'s "report" but only contains system-level metric groups (not associated to any Neuron application)

Regardless of the configuration, the following two JSON objects are always present in the output:

instance_info Contains information about the instance on which neuron-monitor is running.

```
"instance_info": {
  "instance_name": "My_Instance",
  "instance_id": "i-0011223344556677a",
  "instance_type": "inf1.xlarge",
  "instance_availability_zone": "us-west-2b",
  "instance_availability_zone_id": "usw2-az2",
  "instance_region": "us-west-2",
  "ami_id": "ami-0011223344556677b",
  "subnet_id": "subnet-112233ee",
  "error": ""
}
```

Depending on when the instance was launched, the following fields might not be available:

- instance_availability_zone_id : available only for instances launched in 2020-08-24 and later
- instance_region : available only for instances launched on 2020-08-24 and later

- `instance_name` : available only if `instance_region` is set and aws-cli tools are installed

`error` will contain an error string if getting one of the fields, **except those mentioned above**, resulted in an error.

neuron_hardware_info Contains basic information about the Neuron hardware.

```
"neuron_hardware_info": {
  "neuron_device_count": 16,
  "neuroncore_per_device_count": 4,
  "error": ""
}
```

- `neuron_device_count` : number of available Neuron Devices
- `neuroncore_per_device_count` : number of NeuronCores present on each Neuron Device
- `error` : will contain an error string if any occurred when getting this information (usually due to the Neuron Driver not being installed or not running).

Each **metric group** requested in the settings file will get an entry in the resulting output. The general format for such an entry is:

```
"metric_group": {
  "period": 1.015, // Actual captured period, in seconds
  "error": "",     // Error, if any occurred, otherwise an empty string
  [...]            // Metric group specific data
}
```

Neuron application level metric groups

neuroncore_counters

```
"neuroncore_counters": {
  "period": 1.000113182,
  "neuroncores_in_use": {
    "0": {
      "neuroncore_utilization": 42.01,
    },
    "1": {
      "neuroncore_utilization": 42.02,
    },
    "2": {
      "neuroncore_utilization": 42.03,
    },
    "3": {
      "neuroncore_utilization": 42.04,
    }
  },
  "error": ""
}
```

- `"neuroncores_in_use"` is an object containing data for all the NeuronCores that were active when the data was captured, indexed by NeuronCore index: `"neuroncore_index": { neuroncore_data }`
  - `"neuroncore_utilization"` - NeuronCore utilization, in percent, during the captured period

- "error" - string containing any error that occurred when collecting the data

execution_stats

```
"execution_stats": {
  "period": 1.030613214,
  "error_summary": {
    "generic": 0,
    "numerical": 0,
    "transient": 0,
    "model": 0,
    "runtime": 0,
    "hardware": 0
  },
  "execution_summary": {
    "completed": 123,
    "completed_with_err": 0,
    "completed_with_num_err": 0,
    "timed_out": 0,
    "incorrect_input": 0,
    "failed_to_queue": 0
  },
  "latency_stats": {
    "total_latency": {
      "p0": 0.01100001,
      "p1": 0.01100002,
      "p25": 0.01100004,
      "p50": 0.01100008,
      "p75": 0.01100010,
      "p99": 0.01100012,
      "p100": 0.01100013
    },
    "device_latency": {
      "p0": 0.01000001,
      "p1": 0.01000002,
      "p25": 0.01000004,
      "p50": 0.01000008,
      "p75": 0.01000010,
      "p99": 0.01000012,
      "p100": 0.01000013
    }
  },
  "error": ""
},
```

- "error_summary" is an object containing the error counts for the captured period indexed by their type
  - "generic" - generic execution errors
  - "numeric" - NAN errors encountered during execution
  - "transient" - recoverable errors, such as ECC corrections
  - "model" - model-related errors

  - "runtime" - Neuron Runtime errors
  - "hardware" - hardware errors such as uncorrectable ECC issues

- "execution_summary" is an object containing all execution outcome counts for the captured period indexed by their type
  - "completed" - executions completed successfully
  - "completed_with_err" - executions that ended in an error other than a numeric error
  - "completed_with_num_err" - executions that ended in a numeric error
  - "timed_out" - executions that took longer than the Neuron Runtime configured timeout value
  - "incorrect_input" - executions that failed to start due to incorrect input being provided
  - "failed_to_queue" - execution requests that were rejected due to Neuron Runtime not being able to queue them
- "latency_stats" contains two objects containing latency percentiles, in seconds, for the data captured for the model executed during the captured period. If there are no models being executed during this time, the two objects will be null (i.e. "total_latency": null)
  - "total_latency" - percentiles, in seconds, representing

  latency for an execution as measured by the Neuron Runtime - "device_latency" - percentiles, in seconds, representing execution time exclusively on the Neuron Device
- "error" - string containing any error that occurred when collecting the data

memory_used

```
"memory_used": {
  "period": 1.00001,
  "neuron_runtime_used_bytes": {
    "host": 6997643264,
    "neuron_device": 12519788544,
    "usage_breakdown": {
      "host": {
        "application_memory": 6996594688,
        "constants": 0,
        "dma_buffers": 1048576,
        "tensors": 0
      },
      "neuroncore_memory_usage": {
        "0": {
          "constants": 193986816,
          "model_code": 176285056,
          "model_shared_scratchpad": 0,
          "runtime_memory": 0,
          "tensors": 20971520
        },
        "1": {
          "constants": 193986816,
          "model_code": 176285056,
          "model_shared_scratchpad": 0,
          "runtime_memory": 0,
```

(continues on next page)

(continued from previous page)

```
          "tensors": 20971520
        },
        ...
      }
  }
  "loaded_models": [
    {
      "name": "neff",
      "uuid": "91f2f66e83ea419dace1da07617ad39f",
      "model_id": 10005,
      "is_running": false,
      "subgraphs": {
        "sg_00": {
          "memory_used_bytes": {
            "host": 20480,
            "neuron_device": 21001024,
            "usage_breakdown": {
              "host": {
                "application_memory": 20480,
                "constants": 0,
                "dma_buffers": 0,
                "tensors": 0
              },
              "neuron_device": {
                "constants": 20971520,
                "model_code": 29504,
                "runtime_memory": 0,
                "tensors": 0
              }
            }
          },
          "neuroncore_index": 0,
          "neuron_device_index": 12
        }
      }
    },
    ...
    ],
    "error": ""
}
```

- "memory_used" summarizes the amount of memory used by the Neuron application
  - "neuron_runtime_used_bytes" - current amount of memory used by the Neuron application
    - "host" - total host DRAM usage in bytes
    - "neuron_device" - total Neuron device memory usage in bytes
    - "usage_breakdown" - a breakdown of the total memory usage in the other two fields
      - "host" - breakdown of the host memory usage
      - "application_memory" - amount of host memory used by the application - this includes all allocations that are not included in the next categories

- "constants" - amount of host memory used for constants during training (or weights during inference)
- "dma_buffers" - amount of host memory used for DMA transfers
- "tensors" - amount of host memory used for tensors
- "neuroncore_memory_usage" - a breakdown of memory allocated on the Neuron Devices and the NeuronCores for which it was allocated

"0" - "32" (for trn1-32xlarge) - NeuronCores for which the memory was allocated

"constants" - amount of device memory used for constants during training (or weights during inference)

"model_code" - amount of device memory used for models' executable code

"model_shared_scratchpad" - amount of device memory used for the scratchpad shared by the models - a memory region reserved for the models'

internal variables and auxiliary buffers - "runtime_memory" - amount of device memory used by the Neuron Runtime - "tensors" - amount of device memory used for tensors

- "loaded_models" - array containing objects representing loaded models
  - "name" - name of the model
  - "uuid" - unique id for the model
  - "model_id" - Neuron application-assigned ID for this model
  - "is_running" - true if this model is currently started, false otherwise
  - "subgraphs" - object containing all the subgraphs for the model, indexed by their name: "subgraph_name": { subgraph_data }
    - "memory_used_bytes" - memory usage for this subgraph
      - "host" - total host DRAM usage in bytes
      - "neuron_device" - total Neuron device DRAM usage in bytes
      - "usage_breakdown" - a breakdown of memory allocated at load time for this model
      - "host" - breakdown of host memory allocated for this model
      - "application_memory" - amount of host memory allocated for this model by the Neuron Runtime which doesn't fall in any of the next categories
      - "constants" - amount of host memory used for constants during training (or weights during inference)
      - "dma_buffers" - host memory allocated for DMA transfers for this model
      - "tensors" - amount of device memory used for tensors at model load time
      - "neuron_device" - a breakdown of device memory allocated for this model
      - "constants" - amount of device memory used for constants during training (or weights during inference)
      - "model_code" - amount of device memory used for the model's executable code
      - "runtime_memory" - amount of device memory used by the Neuron Runtime for this model
      - "tensors" - amount of device memory allocated for tensors at this model's load time

        * "neuroncore_index" - NeuronCore index on which the subgraph is loaded
        * "neuron_device_index" - Neuron device index on which the subgraph is loaded

- "error" - string containing any error that occurred when collecting the data

neuron_runtime_vcpu_usage

```
"neuron_runtime_vcpu_usage": {
  "period": 1.030604818,
  "vcpu_usage": {
    "user": 42.01,
    "system": 12.34
  },
  "error": ""
}
```

- "vcpu_usage" - object showing vCPU usage in percentages for the Neuron application during the captured period
  - "user" - percentage of time spent in user code by this Neuron Application
  - "system" - percentage of time spent in kernel code by this Neuron application
- "error" - string containing any error that occurred when collecting the data

System level metric groups

neuron_hw_counters

```
"neuron_hw_counters": {
  "period": 1.030359284,
  "neuron_devices": [
    {
      "neuron_device_index": 0,
      "mem_ecc_corrected": 0,
      "mem_ecc_uncorrected": 0,
      "sram_ecc_uncorrected": 0,
      "sram_ecc_corrected": 0
    }
  ],
  "error": ""
},
```

- "neuron_devices" - array containing ECC data for all Neuron devices
  - "neuron_device_index" - Neuron device index
  - "mem_ecc_corrected" - number of corrected ECC events in the Neuron device's DRAM
  - "mem_ecc_uncorrected" - number of uncorrected ECC events in the Neuron device's DRAM
  - "sram_ecc_uncorrected" - number of uncorrected ECC events in the Neuron device's SRAM
  - "sram_ecc_corrected" - number of corrected ECC events in the Neuron device's SRAM
- "error" - string containing any error that occurred when collecting the data

vcpu_usage

```
"vcpu_usage": {
  "period": 0.999974868,
  "average_usage": {
    "user": 32.77,
    "nice": 0,
    "system": 22.87,
    "idle": 39.36,
    "io_wait": 0,
    "irq": 0,
    "soft_irq": 0
  },
  "usage_data": {
    "0": {
      "user": 34.41,
      "nice": 0,
      "system": 27.96,
      "idle": 37.63,
      "io_wait": 0,
      "irq": 0,
      "soft_irq": 0
    },
    "1": {
      "user": 56.84,
      "nice": 0,
      "system": 28.42,
      "idle": 14.74,
      "io_wait": 0,
      "irq": 0,
      "soft_irq": 0
    },
    [...]
  },
  "context_switch_count": 123456,
  "error": ""
}
```

- each vCPU usage object contains the following fields:
  - `"user"` - percentage of time spent in user code
  - `"nice"` - percentage of time spent executing niced user code
  - `"system"` - percentage of time spent executing kernel code
  - `"idle"` - percentage of time spent idle
  - `"io_wait"` - percentage of time spent waiting for IO operations
  - `"irq"` - percentage of time spent servicing hardware interrupts
  - `"soft_irq"` - percentage of time spent servicing software interrupts
- `"average_usage"` - contains the average usage across all vCPUs during the captured period
- `"usage_data"` - contains per vCPU usage during the captured period

- "context_switch_count" - contains the number of vCPU context switches during the captured period
- "error" - string containing any error that occurred when collecting the data

memory_info

```
"memory_info": {
  "period": 5.346411129,
  "memory_total_bytes": 49345835008,
  "memory_used_bytes": 16042344448,
  "swap_total_bytes": 0,
  "swap_used_bytes": 0,
  "error": ""
}
```

- "memory_total_bytes" - total size of the host memory, in bytes
- "memory_used_bytes" - amount of host memory in use, in bytes
- "swap_total_bytes" - total size of the host swap file, in bytes
- "swap_used_bytes" - amount of swap memory in use, in bytes

Companion scripts

neuron-monitor is installed with two example Python companion script: *neuron-monitor-cloudwatch.py* and *neuron-monitor-prometheus.py*.

neuron-monitor-cloudwatch.py

It requires Python3 and the boto3 Python module. It is installed to: `/opt/aws/neuron/bin/neuron-monitor-cloudwatch.py`.

Using neuron-monitor-cloudwatch.py

```
neuron-monitor | neuron-monitor-cloudwatch.py --namespace <namespace> --region <region>
```

For example:

```
neuron-monitor | neuron-monitor-cloudwatch.py --namespace neuron_monitor_test --region␣
↪us-west-2
```

neuron-monitor-prometheus.py

It requires Python3 and the Prometheus client Python module. It is installed to: `/opt/aws/neuron/bin/neuron-monitor-prometheus.py`.

Using neuron-monitor-prometheus.py

```
neuron-monitor | neuron-monitor-prometheus.py --port <port>
```

For example:

```
neuron-monitor | neuron-monitor-prometheus.py --port 8008
```

The default value for `--port` is `8000`.

If your data visualization framework is Grafana, we provided a Grafana dashboard which integrates with Prometheus and this script.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

13.1.2 Neuron Top User Guide

Table of contents

- *Overview*
- *Using neuron-top*
  - *Command line arguments*
  - *User interface*

Overview

`neuron-top` provides useful information about NeuronCore and vCPU utilization, memory usage, loaded models, and Neuron applications.

Note: If you are parsing `neuron-top` output in your automation environment, you can now replace it with `neuron-monitor` (*Neuron Monitor User Guide*) which outputs data in a standardized, easier to parse JSON format.

Using neuron-top

Command line arguments

Launch `neuron-top` by simply typing its name in the shell: `neuron-top`.

User interface

The user interface is divided in 4 sections. The data shown in these sections applies to the currently selected tab - which can be the 'all' tab, which aggregates data from all running Neuron processes, or a tab representing a single Neuron process:

- The `Neuroncore Utilization` section shows the NeuronCore utilization for the currently selected tab
- The `VCPU Utilization` section shows:
  - `System vCPU usage` - the two percentages are user% and system%
  - `Runtime vCPU usage` - same breakdown
- The `Memory Usage Summary` section provides a breakdown of the total memory usage on the Neuron Device as well as on the host:
  - `Host Used Memory` - amount of host memory used by the selected application (or an aggregate of all applications if 'All' is selected)
    - `Total` - total amount of host memory used

    * `Tensors` - amount of host memory used for tensors
    * `Constants` - amount of host memory used for constants (for applications running training) or weights (for applications running inferences)
    * `DMA Buffers` - amount of host memory used for DMA transfers
    * `App. Memory` - amount of host memory used by the application that doesn't fall in any of the previous categories
  - `Device Used Memory` - amount of device memory used by the selected application (or an aggregate of all applications if 'All' is selected)
    * `Total` - total amount of device memory used
    * `Tensors` - amount of device memory used for tensors
    * `Constants` - amount of device memory used for constants (for applications running training) or weights (for applications running inferences)
    * `Model Code` - amount of device memory used for storing model executable code
    * `Runtime Memory` - amount of device memory used by the Neuron Runtime (outside of the previous categories)
    * `Model Scratchpad` - amount of device memory used for the shared model scratchpad, a shared buffer used for internal model variables and other auxiliary buffers

- `Memory Usage Details` contains memory usage data organized as a tree which can be expanded/collapsed. The columns are:
  - `Model ID` - the Neuron Runtime identifier for this model instance
  - `Host Memory` - amount of host memory used
  - `Device Memory` - amount of device memory used

The tree view shows the amount of memory used for the same categories shown in the `Memory Usage Summary` but in this section they are attached to either a model (if the memory has been allocated at model load time for that model), or to a NeuronCore (if the memory can't be associated with a model, but has been allocated for that NeuronCore). The 'parent' shows the total amount of memory used - the sum of its children.

Note: The up/down/left/right keys can be used to navigate the tree view. The 'x' key expands/collapses the entire tree.

The bottom bar shows which Neuron process' data is currently displayed by highlighting its tag using a green font and marking it using a pair of '>', '<' characters. The 'all' tab shows an aggregated view of all the Neuron processes currently running on the instance.

Note: The '1'-'9' keys select the current tab.'a'/'d' selects the previous/next tab on the bar.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

13.1.3 Neuron LS User Guide

To identify number of Neuron Devices in a given instance use the `neuron-ls` command. `neuron-ls` will also show which processes are using each Device, including the command used to launch each of those processes.

neuron-ls CLI

neuron-ls [options]

Available options:

- `--wide, -w`: Displays the table in a wider format.
- `--show-all-procs, -a`: Show all processes using the Neuron Devices, including processes that aren't using Neuron Runtime 2.x such as `neuron-monitor` or `neuron-ls` itself.
- `--topology, -t`: Display topology information about the system's Neuron Devices.

Note: If you're running Neuron Runtime 1.x `neuron-rtd` daemons, use `sudo` with the `neuron-ls` command to obtain their address and version information. Without `sudo`, the `neuron-rtd` processes will be considered generic processes using one or more Devices and will only be displayed when running `neuron-ls` with the `--show-all-procs` option.

Examples

First we will show the output of `neuron-ls` on an Inf1.6xlarge instance.

```
$ neuron-ls
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| NEURON | NEURON | NEURON | CONNECTED |     PCI      |  PID  |                 COMMAND ␣
↪                | RUNTIME |
| DEVICE | CORES  | MEMORY |  DEVICES  |     BDF      |       |                         ␣
↪                | VERSION |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 0      | 4      | 8 GB   | 1         | 0000:00:1c.0 | 23518 | neuron-app01 infer --
↪input-data-direc... | 2.0.0   |
|        |        |        |           |              | 23531 | neuron-app02 infer --
↪input-data-direc... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 1      | 4      | 8 GB   | 2, 0      | 0000:00:1d.0 | 23595 | neuron-app01 infer --
↪input-data-direc... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 2      | 4      | 8 GB   | 3, 1      | 0000:00:1e.0 | 23608 | neuron-app02 infer --
↪input-data-direc... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 3      | 4      | 8 GB   | 2         | 0000:00:1f.0 | NA    | NA                      ␣
↪                | NA      |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
```

(continues on next page)

(continued from previous page)

```
$ neuron-ls --wide
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+
| NEURON | NEURON | NEURON | CONNECTED |     PCI      |  PID  |                         ␣
↪           COMMAND                                     | RUNTIME |
| DEVICE | CORES  | MEMORY |  DEVICES  |     BDF      |       |                         ␣
↪                                                       | VERSION |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+
| 0      | 4      | 8 GB   | 1         | 0000:00:1c.0 | 23518 | neuron-app01 infer --
↪input-data-directory ~/my_input_data --inference-count 5... | 2.0.0   |
|        |        |        |           |              | 23531 | neuron-app02 infer --
↪input-data-directory ~/my_input_data --inference-count 5... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+
| 1      | 4      | 8 GB   | 2, 0      | 0000:00:1d.0 | 23595 | neuron-app01 infer --
↪input-data-directory ~/my_input_data --inference-count 5... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+
| 2      | 4      | 8 GB   | 3, 1      | 0000:00:1e.0 | 23608 | neuron-app02 infer --
↪input-data-directory ~/my_input_data --inference-count 5... | 2.0.0   |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+
| 3      | 4      | 8 GB   | 2         | 0000:00:1f.0 | NA    | NA                      ␣
↪                                                       | NA      |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪-------------------------------------------------------+---------+

$ neuron-ls --show-all-procs
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| NEURON | NEURON | NEURON | CONNECTED |     PCI      |  PID  |                 COMMAND ␣
↪                | RUNTIME |
| DEVICE | CORES  | MEMORY |  DEVICES  |     BDF      |       |                         ␣
↪                | VERSION |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 0      | 4      | 8 GB   | 1         | 0000:00:1c.0 | 23518 | neuron-app01 infer --
↪input-data-direc... | 2.0.0   |
|        |        |        |           |              | 23531 | neuron-app02 infer --
↪input-data-direc... | 2.0.0   |
|        |        |        |           |              | 23764 | neuron-monitor          ␣
↪                | NA      |
|        |        |        |           |              | 23829 | neuron-ls --show-all-
↪procs               | NA      |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 1      | 4      | 8 GB   | 2, 0      | 0000:00:1d.0 | 23595 | neuron-app01 infer --
↪input-data-direc... | 2.0.0   |
|        |        |        |           |              | 23764 | neuron-monitor          ␣
↪                | NA      |
|        |        |        |           |              | 23829 | neuron-ls --show-all-
↪procs               | NA      |
```

(continues on next page)

(continued from previous page)

```
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 2      | 4      | 8 GB   | 3, 1      | 0000:00:1e.0 | 23608 | neuron-app02 infer --
↪input-data-direc... | 2.0.0   |
|        |        |        |           |              | 23764 | neuron-monitor         ␣
↪                | NA      |
|        |        |        |           |              | 23829 | neuron-ls --show-all-
↪procs               | NA      |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+
| 3      | 4      | 8 GB   | 2         | 0000:00:1f.0 | 23764 | neuron-monitor         ␣
↪                | NA      |
|        |        |        |           |              | 23829 | neuron-ls --show-all-
↪procs               | NA      |
+--------+--------+--------+-----------+--------------+-------+-------------------------
↪----------------+---------+

  $ neuron-ls --topology
+--------+--------+--------+-----------+---------+
| NEURON | NEURON | NEURON | CONNECTED |   PCI   |
| DEVICE | CORES  | MEMORY |  DEVICES  |   BDF   |
+--------+--------+--------+-----------+---------+
| 0      | 4      | 8 GB   | 1         | 00:1c.0 |
| 1      | 4      | 8 GB   | 2, 0      | 00:1d.0 |
| 2      | 4      | 8 GB   | 3, 1      | 00:1e.0 |
| 3      | 4      | 8 GB   | 2         | 00:1f.0 |
+--------+--------+--------+-----------+---------+

Neuron Device Topology


  [ 0 ]--[ 1 ]--[ 2 ]--[ 3 ]
```

On Trn1 instances `neuron-ls` works similarly. Below is an example displaying the topology for a Trn1.32xlarge instance.

```
$ neuron-ls --topology
+--------+--------+--------+---------------+---------+
| NEURON | NEURON | NEURON |   CONNECTED   |   PCI   |
| DEVICE | CORES  | MEMORY |    DEVICES    |   BDF   |
+--------+--------+--------+---------------+---------+
| 0      | 2      | 32 GB  | 12, 3, 4, 1   | 00:04.0 |
| 1      | 2      | 32 GB  | 13, 0, 5, 2   | 00:05.0 |
| 2      | 2      | 32 GB  | 14, 1, 6, 3   | 00:06.0 |
| 3      | 2      | 32 GB  | 15, 2, 7, 0   | 00:07.0 |
| 4      | 2      | 32 GB  | 0, 7, 8, 5    | 00:08.0 |
| 5      | 2      | 32 GB  | 1, 4, 9, 6    | 00:09.0 |
| 6      | 2      | 32 GB  | 2, 5, 10, 7   | 00:0a.0 |
| 7      | 2      | 32 GB  | 3, 6, 11, 4   | 00:0b.0 |
| 8      | 2      | 32 GB  | 4, 11, 12, 9  | 00:0c.0 |
| 9      | 2      | 32 GB  | 5, 8, 13, 10  | 00:0d.0 |
| 10     | 2      | 32 GB  | 6, 9, 14, 11  | 00:0e.0 |
```

(continues on next page)

(continued from previous page)

```
| 11     | 2      | 32 GB  | 7, 10, 15, 8  | 00:0f.0 |
| 12     | 2      | 32 GB  | 8, 15, 0, 13  | 00:10.0 |
| 13     | 2      | 32 GB  | 9, 12, 1, 14  | 00:11.0 |
| 14     | 2      | 32 GB  | 10, 13, 2, 15 | 00:12.0 |
| 15     | 2      | 32 GB  | 11, 14, 3, 12 | 00:13.0 |
+--------+--------+--------+---------------+---------+
Neuron Device Topology
      *        *        *        *
      |        |        |        |

*--[ 0 ]--[ 1 ]--[ 2 ]--[ 3 ]--*

      |        |        |        |

*--[ 4 ]--[ 5 ]--[ 6 ]--[ 7 ]--*

      |        |        |        |

*--[ 8 ]--[ 9 ]--[10 ]--[11 ]--*

      |        |        |        |

*--[12 ]--[13 ]--[14 ]--[15 ]--*

      |        |        |        |
      *        *        *        *
```

- NEURON DEVICE: Logical ID assigned to the Neuron Device.
- NEURON CORES: Number of NeuronCores present in the Neuron Device.
- NEURON MEMORY: Amount DRAM memory in Neuron Device.
- CONNECTED DEVICES: Logical ID of Neuron Devices connected to this Neuron Device.
- PCI BDF: PCI Bus Device Function (BDF) ID of the device.
- PID: ID of the process using this NeuronDevice.
- COMMAND: Command used to launch the process using this Neuron Device.
- RUNTIME VERSION: Version of Neuron Runtime (if applicable) for the application using this Neuron Device.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

13.1.4 Neuron System Tools

Table of Contents

- *Neuron Tools [2.6.0.0]*
- *Neuron Tools [2.5.19.0]*
- *Neuron Tools [2.5.16.0]*
- *Neuron Tools [2.4.6.0]*
- *Neuron Tools [2.1.4.0]*
- *Neuron Tools [2.0.790.0]*
- *Neuron Tools [2.0.623.0]*
- *Neuron Tools [2.0.494.0]*
- *Neuron Tools [2.0.327.0]*
- *Neuron Tools [2.0.277.0]*

Neuron Tools [2.6.0.0]

Date: 12/09/2022

This release adds support for profiling with the Neuron Plugin for TensorBoard on TRN1. Please check out the documentation *Neuron Plugin for TensorBoard (Trn1)*.

New in the release:

- Updated profile post-processing for workloads executed on TRN1

Neuron Tools [2.5.19.0]

Date: 11/07/2022

New in the release:

- Minor bug fixes and improvements.

Neuron Tools [2.5.16.0]

Date: 10/26/2022

New in the release:

- New `neuron-monitor` and `neuron-top` feature: **memory utilization breakdown**. This new feature provides more details on how memory is being currently used on the Neuron Devices as well as on the host instance.
- `neuron-top`'s UI layout has been updated to accommodate the new **memory utilization breakdown** feature.
- `neuron-monitor`'s `inference_stats` metric group was renamed to `execution_stats`. While the previous release still supported `inference_stats`, starting this release the name `inference_stats` is considered deprecated and can't be used anymore.

Note: For more details on the new **memory utilization breakdown** feature in `neuron-monitor` and `neuron-top` check out the full user guides: *Neuron Monitor User Guide* and *Neuron Top User Guide*.

Bug Fixes:

- Fix a rare crash in `neuron-top` when the instance is under heavy CPU load.
- Fix process names on the bottom tab bar of `neuron-top` sometimes disappearing for smaller terminal window sizes.

Neuron Tools [2.4.6.0]

Date: 10/10/2022

This release adds support for both EC2 INF1 and TRN1 platforms. Name of the package changed from aws-neuron-tools to aws-neuronx-tools. Please remove the old package before installing the new one.

New in the release:

- Added support for ECC counters on Trn1
- Added version number output to neuron-top
- Expanded support for longer process tags in neuron-monitor.
- Removed hardware counters from the default neuron-monitor config to avoid sending repeated errors - will add back in future release.
- `neuron-ls` - Added option `neuron-ls --topology` with ASCII graphics output showing the connectivity between Neuron Devices on an instance. This feature aims to help in understanding pathways between Neuron Devices and in exploiting code or data locality.

Bug Fixes:

- Fix neuron-monitor and neuron-top to show the correct Neuron Device when running in a container where not all devices are present.

Neuron Tools [2.1.4.0]

Date: 04/29/2022

- Minor updates

Neuron Tools [2.0.790.0]

Date: 03/25/2022

- `neuron-monitor`: fixed a floating point error when calculating CPU utilization.

Neuron Tools [2.0.623.0]

Date: 01/20/2022

New in the release:

- `neuron-top` - Added "all" tab that aggregates all aggregate all running Neuron processes into a single view.
- `neuron-top` - Improved startup time to approximately 1.5 seconds in most cases.
- `neuron-ls` - Removed header message about updating tools from neuron-ls output

Bug fixes:

- `neuron-top` - Reduced single CPU core usage down to 0.7% from 80% on inf1.xlarge when running `neuron-top` by switching to an event-driven approach for screen updates.

Neuron Tools [2.0.494.0]

Date: 12/27/2021

- Security related updates related to log4j vulnerabilities.

Neuron Tools [2.0.327.0]

Date: 11/05/2021

- Updated Neuron Runtime (which is integrated within this package) to `libnrt 2.2.18.0` to fix a container issue that was preventing the use of containers when /dev/neuron0 was not present. See details here neuron-runtime-release-notes.

Neuron Tools [2.0.277.0]

Date: 10/27/2021

New in this release:

- Tools now support applications built with Neuron Runtime 2.x (`libnrt.so`).

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.
  - Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

- Updates have been made to `neuron-ls` and `neuron-top` to significantly improve the interface and utility of information provided.
- Expands `neuron-monitor` to include additional information when used to monitor latest Frameworks released with Neuron 1.16.0.

  neuron_hardware_info Contains basic information about the Neuron hardware.

```
"neuron_hardware_info": {
   "neuron_device_count": 16,
   "neuroncore_per_device_count": 4,
   "error": ""
}
```

  - `neuron_device_count` : number of available Neuron Devices
  - `neuroncore_per_device_count` : number of NeuronCores present on each Neuron Device
  - `error` : will contain an error string if any occurred when getting this information (usually due to the Neuron Driver not being installed or not running).

- `neuron-cli` entering maintenance mode as it's use is no longer relevant when using ML Frameworks with an integrated Neuron Runtime (libnrt.so). see maintenance_mxnet_1_5 for more information.
- For more information visit *Neuron Tools*

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

System Tools

- *Neuron Monitor User Guide*
- *Neuron Top User Guide*
- *Neuron LS User Guide*
- *What's New*

This document is relevant for: `Inf1`, `Trn1`

13.2 TensorBoard

13.2.1 TensorBoard for Trn1

This document is relevant for: `Trn1`

Track Training Progress in TensorBoard using PyTorch Neuron

Table of Contents

- *Multi-layer perceptron MNIST model*
- *Output TensorBoard logs*
- *View loss in TensorBoard*

This tutorial explains how to track training progress in TensorBoard while running a multi-layer perceptron MNIST model on Trainium using PyTorch Neuron.

Multi-layer perceptron MNIST model

This tutorial is based on the MNIST example for PyTorch Neuron on Trainium. For the full tutorial, please see *Multi-Layer Perceptron Training Tutorial*.

Output TensorBoard logs

To generate TensorBoard logs, we first modify the training script to use the `SummaryWriter`:

```
from torch.utils.tensorboard import SummaryWriter
writer = SummaryWriter('./output')
```

In the training loop, we can then use the `add_scalar` API to log the loss per step.

```
writer.add_scalar("step loss", loss, idx)
```

At the end of the script, add `writer.flush()` to ensure all logs are written.

Save the following code as `train_tb.py` and run it as `python3 train_tb.py` on a Trn1 instance. The generated logs can be found in the `./output` directory that was passed to `SummaryWriter`.

```
import os
import time
import torch
import torch.nn as nn
import torch.nn.functional as F

from torchvision.datasets import mnist
from torch.optim import SGD
from torch.utils.data import DataLoader
from torchvision.transforms import ToTensor

# XLA imports
import torch_xla.core.xla_model as xm

from torch.utils.tensorboard import SummaryWriter

# Declare 3-layer MLP for MNIST dataset
class MLP(nn.Module):
  def __init__(self, input_size = 28 * 28, output_size = 10, layers = [120, 84]):
      super(MLP, self).__init__()
      self.fc1 = nn.Linear(input_size, layers[0])
      self.fc2 = nn.Linear(layers[0], layers[1])
      self.fc3 = nn.Linear(layers[1], output_size)

  def forward(self, x):
      x = F.relu(self.fc1(x))
      x = F.relu(self.fc2(x))
      x = self.fc3(x)
      return F.log_softmax(x, dim=1)

# Load MNIST train dataset
train_dataset = mnist.MNIST(root='./MNIST_DATA_train', \
```

(continues on next page)

(continued from previous page)

```
                                train=True, download=True, transform=ToTensor())

def main():
    # Prepare data loader
    train_loader = DataLoader(train_dataset, batch_size=32)

    # Fix the random number generator seeds for reproducibility
    torch.manual_seed(0)

    # XLA: Specify XLA device (defaults to a NeuronCore on Trn1 instance)
    device = 'xla'

    # Move model to device and declare optimizer and loss function
    model = MLP().to(device)
    optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
    loss_fn = torch.nn.NLLLoss()

    # Use SummaryWriter to generate logs for TensorBoard
    writer = SummaryWriter('./output')

    # Run the training loop
    print('----------Training ---------------')
    model.train()
    start = time.time()
    for idx, (train_x, train_label) in enumerate(train_loader):
        optimizer.zero_grad()
        train_x = train_x.view(train_x.size(0), -1)
        train_x = train_x.to(device)
        train_label = train_label.to(device)
        output = model(train_x)
        loss = loss_fn(output, train_label)
        writer.add_scalar("step loss", loss, idx) # add the step loss to the TensorBoard␣
↪logs
        loss.backward()
        optimizer.step()
        xm.mark_step() # XLA: collect ops and run them in XLA runtime
        if idx < 2: # skip warmup iterations
            start = time.time()

    # Compute statistics
    interval = idx - 2 # skip warmup iterations
    throughput = interval / (time.time() - start)
    print("Train throughput (iter/sec): {}".format(throughput))
    print("Final loss is {:0.4f}".format(loss.detach().to('cpu')))

    # Ensure TensorBoard logs are all written
    writer.flush()

    # Save checkpoint for evaluation
    os.makedirs("checkpoints", exist_ok=True)
    checkpoint = {'state_dict': model.state_dict()}
    # XLA: use xm.save instead of torch.save to ensure states are moved back to cpu
```

(continues on next page)

(continued from previous page)

```
    # This can prevent "XRT memory handle not found" at end of test.py execution
    xm.save(checkpoint,'checkpoints/checkpoint.pt')

    print('----------End Training ---------------')

if __name__ == '__main__':
    main()
```

View loss in TensorBoard

In order to view your training metrics, install TensorBoard in your Python environment:

```
pip install tensorboard
```

Then, launch TensorBoard with the `./output` directory

```
tensorboard --logdir ./output
```

Once running, open a new SSH connection to the instance and port-forward TCP port 6006 (ex: -L 6006:127.0.0.1:6006). Once the tunnel is established, TensorBoard can then be accessed via web browser at the following URL: http://localhost:6006. Please note that you will not be able to access TensorBoard if you disconnect your port-forwarding SSH session to the Trainium instance.

In TensorBoard, you can now see the loss per step plotted. When capturing loss for multiple runs, you can plot them together on the same graph to compare runs. Be sure to change the output directory for different runs, for example `./output/run1` for the first, `./output/run2` for the second, etc.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

Neuron Plugin for TensorBoard (Trn1)

Table of Contents

- *Overview*
- *Enable profiling on Trn1*
- *Launch TensorBoard*
- *View results in TensorBoard*
- *Neuron Trace View*
- *Neuron Operator View*
- *Neuron Operator Timeline View*

Overview

This guide is for developers who want to better understand how their model is executed using Neuron SDK through TensorBoard.

The Neuron plugin for TensorBoard provides metrics to the performance of machine learning tasks accelerated using the Neuron SDK. It is compatible with TensorBoard versions 1.15 and higher. It provides visualizations and profiling results for graphs executed on NeuronCores.

Note: The following information is compatible with Neuron SDK for Trn1. For a walkthrough on Inf1, please check out the guide *Neuron Plugin for TensorBoard (Inf1)*.

Enable profiling on Trn1

Note: Profiling is currently only supported with PyTorch Neuron (`torch-neuronx`).

Please refer to the following guides:

- **PyTorch-Neuron**
  - torch-neuronx-profiling-with-tb

Launch TensorBoard

In this step, we will process the Neuron profile data and launch TensorBoard.

1. Install the Neuron plugin for Tensorboard on your EC2 instance.

```
python -m pip config set global.extra-index-url "https://pip.repos.neuron.amazonaws.com"

pip install tensorboard-plugin-neuronx
```

Note: If using TensorBoard >= 2.5, please use the `--load_fast=false` option when launching. `tensorboard --logdir results --load_fast=false`

2. After you see the following message, TensorBoard is ready to use. By default, TensorBoard will be launched at `localhost:6006`.

```
...
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_
↪all
TensorBoard 2.4.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

View results in TensorBoard

In this step, we will view the Neuron plugin for TensorBoard from a browser on your local development machine.

1. Connect to the EC2 instance where TensorBoard is running while enabling port forwarding. In this example, we assume TensorBoard has been launched using the default address `localhost:6006`.

```
# if Ubuntu-based AMI
ssh -i <PEM key file> ubuntu@<instance DNS> -L 6006:localhost:6006

# if AL2-based AMI
ssh -i <PEM key file> ec2-user@<instance DNS> -L 6006:localhost:6006
```

2. In a browser, visit .

3. In the top navigation bar, switch from `Graphs` to `Neuron`. If it does not show up, please wait a while and refresh the page while the plugin loads. If the issue persists, check the `Inactive` dropdown list on the right and check for `Neuron`.

4. If TensorBoard failed to find the generated logs, you will see the following message:

In this case, please make sure the version of the `aws-neuronx-tools` package and the Neuron framework package is

from Neuron release 2.6 or newer.

Neuron Trace View

The trace view gives a high level timeline of execution by aligning Neuron events, such as Neuron Device execution, data transfers, and Collective Compute synchronization (if applicable), with other events from the XLA profiler.

Use this view to better understand bottlenecks during the run, and potentially experiment with how execution changes by moving the `mark_step()` call which will execute the graph.

Neuron Operator View

The operator view can show timing information for both the framework operators and HLO operators by selecting the `operator-framework` and `operator-hlo` tools respectively. The pie charts show breakdowns of the time taken by device, as well as per operator on a single device. The table below lists out the operators and can be sorted by clicking on the columnn headers. For fused operations, hover over the `?` to see which operators are being executed.

For a quick glance at the most time consuming operators, click the `Time %` column in the table to sort by the relative time spent on this type of operation compared to the rest of the model.

Neuron Operator Timeline View

The operator timeline view is a detailed look into a single execution with Neuron. Compute and data transfers are shown as two separate processes, with each showing a hierarchical view of the framework operators and their correspondingg HLO operation. The fused operations can be a result of compiler optimizations or are operations that are running in parallel on the device. Each bar can be clicked to show information regarding which operators are overlapped.

This view can give better insight into how operators translate to Neuron, as well as how certain Neuron compiler options may improve performance.

This document is relevant for: `Trn1`

This document is relevant for: `Inf1`, `Trn1`

Neuron Plugin for TensorBoard Release Notes

Table of Contents

- *Known Issues and Limitations - Updated 11/29/2022*
- *Neuron Plugin for TensorBoard release [2.5.0.0]*
- *Neuron Plugin for TensorBoard release [2.4.0.0]*
- *Neuron Plugin for TensorBoard release [2.3.0.0]*
- *Neuron Plugin for TensorBoard release [2.2.0.0]*
- *[2.1.2.0]*
- *[2.1.0.0]*
- *[2.0.29.0]*
- *[1.15.0.1.2.6.0]*

- *[1.15.0.1.1.1.0]*
- *[1.15.0.1.0.615.0]*
- *[1.15.0.1.0.600.0]*
- *[1.15.0.1.0.570.0]*
- *[1.15.0.1.0.513.0]*
- *[1.15.0.1.0.491.0]*
- *[1.15.0.1.0.466.0]*
- *[1.15.0.1.0.392.0]*
- *[1.15.0.1.0.366.0]*
- *[1.15.0.1.0.315.0]*
- *[1.15.0.1.0.306.0]*
- *[1.15.0.1.0.280.0]*

Known Issues and Limitations - Updated 11/29/2022

The following are not limitations in the Neuron plugin, but may affect your ability to use TensorBoard.

- The Neuron plugin for Trn1 (`tensorboard-plugin-neuronx`) is not compatible with the Neuron plugin for Inf1 (`tensorboard-plugin-neuron`). Please ensure you only have only the correct package installed.

Neuron Plugin for TensorBoard release [2.5.0.0]

Date: 12/09/2022

Summary

- Added support for PyTorch Neuron on Trn1 (`torch-neuronx`) with new views! Includes a trace view, an operator view, and an operator timeline view. For more info, check out the documentation *Neuron Plugin for TensorBoard (Trn1)*.

  Important:

  - You must update to the latest Neuron Tools (`aws-neuronx-tools` version 2.6 or newer) and install `tensorboard-plugin-neuronx` for proper functionality of the Neuron plugin on Trn1.
  - For Inf1, please continue to use `tensorboard-plugin-neuron`. Refer to the getting started guide on Inf1 *Neuron Plugin for TensorBoard (Inf1)*.

Neuron Plugin for TensorBoard release [2.4.0.0]

Date: 04/29/2022

Summary

- Minor updates.

Neuron Plugin for TensorBoard release [2.3.0.0]

Date: 03/25/2022

Summary

- Minor updates.

Neuron Plugin for TensorBoard release [2.2.0.0]

Date: 10/27/2021

New in this release

- Neuron Plugin for TensorBoard now support applications built with Neuron Runtime 2.x (`libnrt.so`).

  Important:

  - You must update to the latest Neuron Driver (`aws-neuron-dkms` version 2.1 or newer) for proper functionality of the new runtime library.
  - Read *Introducing Neuron Runtime 2.x (libnrt.so)* application note that describes *why are we making this change* and how *this change will affect the Neuron SDK* in detail.
  - Read *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for detailed information of how to migrate your application.

[2.1.2.0]

Date: 8/12/2021

Summary

- Adds support for Neuron Tensorflow 2.5+

[2.1.0.0]

Date: 5/28/2021

Summary

- No major changes or fixes. Released with other Neuron packages.

[2.0.29.0]

Date: 4/30/2021

Summary

- First release Neuron plugin for TensorBoard. Check out it out here: *Neuron Plugin for TensorBoard (Inf1)*.
  - The Neuron plugin is now compatible with TensorBoard 2.0 and higher, in addition to TensorBoard 1.15
  - Provides a centralized place to better understand execution using Neuron SDK.
  - Continues support visualization for TensorFlow graphs, with support for PyTorch and MXNet coming in future releases.
- Neuron plugin for TensorBoard is supported for Neuron tools >= 1.5, which is first introduced in Neuron v1.13.0 release
- TensorBoard-Neuron is deprecated, and only supported for Neuron tools <= 1.4.12.0. The final version, 1.4.12.0 is part of Neuron v1.12.2 release.

[1.15.0.1.2.6.0]

Date: 2/24/2021

Summary

- Fix for CVE-2021-3177.

[1.15.0.1.1.1.0]

Date: 12/23/2020

Summary

- Minor internal improvements.

[1.15.0.1.0.615.0]

Date: 11/17/2020

Summary

- Fix issue with viewing chrome trace in Neuron profile plugin in Chrome 80+.

Resolved Issues

- Updated dependencies to polyfill missing APIs used by chrome trace in newer browser versions.

[1.15.0.1.0.600.0]

Date: 09/22/2020

Summary

- Minor internal improvements.

[1.15.0.1.0.570.0]

Date: 08/08/2020

Summary

- Minor internal improvements.

[1.15.0.1.0.513.0]

Date: 07/16/2020

Summary

- Minor internal improvements.

[1.15.0.1.0.491.0]

Date 6/11/2020

Summary

Fix issue where utilization was missing in the op-profile view.

Resolved Issues

- The op-profile view in the Neuron Profile plugin now correctly shows the overall NeuronCore utilization.

[1.15.0.1.0.466.0]

Date 5/11/2020

Summary

Fix potential installation issue when installing both tensorboard and tensorboard-neuron.

Resolved Issues

- Added tensorboard as a dependency in tensorboard-neuron. This prevents the issue of overwriting tensorboard-neuron features when tensorboard is installed after tensorboard-neuron.

Other Notes

[1.15.0.1.0.392.0]

Date 3/26/2020

Summary

Added ability to view CPU node latency in the Graphs plugin and the Neuron Profile plugins.

Major New Features

- Added an aggregate view in addition to the current Neuron subgraph view for both the Graphs plugin and the Neuron Profile plugin.
- When visualizing a graph executed on a Neuron device, CPU node latencies are available when coloring the graph by "Compute time" using the "neuron_profile" tag.
- The Neuron Profile plugin now has an overview page to compare time spent on Neuron device versus on CPU.

Other Notes

- Requires Neuron-RTD config option "enable_node_profiling" to be set to "true"

[1.15.0.1.0.366.0]

Date 02/27/2020

Summary

Reduced load times and fixed crashes when loading large models for visualization.

Resolved Issues

- Enable large attribute filtering by default
- Reduced load time for graphs with attributes larger than 1 KB
- Fixed a fail to load graphs with many large attributes totaling more than 1 GB in size

[1.15.0.1.0.315.0]

Date 12/20/2019

Summary

No major chages or fixes. Released with other Neuron packages.

[1.15.0.1.0.306.0]

Date 12/1/2019

Summary

Major New Features

Resolved Issues

Known Issues & Limits

Same as prior release

Other Notes

[1.15.0.1.0.280.0]

Date 11/29/2019

Summary

Initial release packaged with DLAMI.

Major New Features

N/A, initial release.

See user guide here: https://github.com/aws/aws-neuron-sdk/blob/master/docs/neuron-tools/getting-started-tensorboard-neuron.md

Resolved Issues

N/A - first release

Known Issues & Limits

- Must install TensorBoard-Neuron by itself, or after regular TensorBoard is installed. If regular Tensorboard is installed after TensorBoard-Neuron, it may overwrite some needed files.
- Utilization missing in Op Profile due to missing FLOPs calculation (see overview page instead)
- Neuron Profile plugin may not immediately show up on launch (try reloading the page)
- Graphs with NeuronOps may take a long time to load due to attribute size
- Instructions that cannot be matched to a framework layer/operator name show as “” (blank)
- CPU Usage section in chrome-trace is not applicable

- Debugger currently supports TensorFlow only
- Visualization requires a TensorFlow-compatible graph

Other Notes

This document is relevant for: `Inf1`, `Trn1`

13.2.2 TensorBoard for Inf1

This document is relevant for: `Inf1`

Neuron Plugin for TensorBoard (Inf1)

Table of Contents

- *Overview*
- *Compile the neural network*
- *Enable profiling*
- *Launch TensorBoard*
- *View results in TensorBoard*
- *Visualize graphs executed on Neuron*
  - *Show how the graph was partition to run on NeuronCores*
  - *Inspect which operators consumes the most time*
  - *Check out Neuron support operators for each framework*
  - *Filter view by device*
  - *Expand/collapse subgraphs and view operator details*
- *Viewing the Neuron profile data*
  - *See performance summary*
  - *Get a breakdown of time spent per NeuronCore*
  - *Get a breakdown of time spent per operator*

Overview

This guide is for developers who want to better understand how their model is executed using Neuron SDK through TensorBoard.

The Neuron plugin for TensorBoard provides metrics to the performance of machine learning tasks accelerated using the Neuron SDK. It is compatible with TensorBoard versions 1.15 and higher. It provides visualizations and profiling results for graphs executed on NeuronCores.

Note: The following information is compatible with Neuron SDK for Inf1. For a walkthrough on the latest version, please check out the guide *Neuron Plugin for TensorBoard (Trn1)*.

Note: Graph visualization is currently only supported for TensorFlow-Neuron. Support for MXNet-Neuron and PyTorch-Neuron visualization will be added in a future release.

Compile the neural network

3. Refer to the following guides on how to compile a graph using Neuron SDK.

- **TensorFlow-Neuron**
  - *Running ResNet50 on Inferentia*
- **PyTorch-Neuron:**
  - "Compile model for Neuron" in *PyTorch-Neuron Resnet50 Tutorial*
- **MXNet-Neuron:**
  - *Running Neuron Apache MXNet (Incubating) ResNet50 on Inferentia*

Enable profiling

In this step, we enable Neuron profile data collection and collect results from executing an inference.

4.1. To start profiling the neural network and collect inference traces, create a directory where profile data will be dumped and set the `NEURON_PROFILE` environment variable. In this example, we will assume this directory is `$HOME/profile`

```
mkdir -p $HOME/profile
export NEURON_PROFILE=$HOME/profile
```

4.2. Ensure Neuron Tools are executable by setting the `PATH` environment variable.

```
export PATH=/opt/aws/neuron/bin:$PATH
```

4.3. Execute inference!

Note: Please run the inference script outside of Jupyter notebook. Profiling in Jupyter notebook is not supported at this time.

Note: Please ensure the inference script executes only one inference, as profiling results are currently only supported for a single inference.

For more info on how to execute inference, refer to the following guides:

- **TensorFlow-Neuron**
  - *Running ResNet50 on Inferentia*

- **PyTorch-Neuron**
  - "Run inference on Single Core" in *ResNet50 model for Inferentia*
- **MXNet-Neuron**
  - *Running Neuron Apache MXNet (Incubating) ResNet50 on Inferentia*

4.4. Check if profiling results were successfully saved. In the directory pointed to by `NEURON_PROFILE` environment variable set in Step 4.1, there should be at least two files, one with the `.neff` extension and one with the `.ntff` extension. For TensorFlow-Neuron users, the graph file (`.pb`) will also be in this directory.

```
ls $NEURON_PROFILE
```

Launch TensorBoard

In this step, we will process the Neuron profile data and launch TensorBoard.

5.1. Install the Neuron plugin for Tensorboard.

If you are using the DLAMI TensorFlow-Neuron Conda environment, please run the following to update TensorBoard before installing the Neuron plugin.

```
pip install "tensorboard<=2.4.0" --force-reinstall
```

Modify Pip repository configurations to point to the Neuron repository:

```
tee $VIRTUAL_ENV/pip.conf > /dev/null <<EOF
[global]
extra-index-url = https://pip.repos.neuron.amazonaws.com
EOF
```

```
pip install tensorboard-plugin-neuron
```

5.2. After collecting the raw profile data, we need to post-process it to create the log files used by the Neuron plugin. This can be done when launching TensorBoard by passing an extra flag `--run_neuron_profiler`. Using this flag will create the directory specified by `--logdir` and populate it with Neuron plugin data. Please note that the `NEURON_PROFILE` environment variable set in Step 4.1 must still point to the same directory as before.

```
tensorboard --logdir results --run_neuron_profiler
```

Note: If using TensorBoard >= 2.5, please use the `--load_fast=false` option when launching. `tensorboard --logdir results --run_neuron_profiler --load_fast=false`

5.3. After you see the following message, TensorBoard is ready to use. By default, TensorBoard will be launched at `localhost:6006` on the Deployment Instance.

```
...
Running neuron-profile
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_
↪all
TensorBoard 2.4.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

View results in TensorBoard

In this step, we will view the Neuron plugin for TensorBoard from a browser on your local development machine.

6.1. Connect to the Deployment Instance while enabling port forwarding. In this example, we assume TensorBoard has been launched using the default address `localhost:6006` on the Deployment Instance.

```
# if Ubuntu-based AMI
ssh -i <PEM key file> ubuntu@<instance DNS> -L 6006:localhost:6006

# if AL2-based AMI
ssh -i <PEM key file> ec2-user@<instance DNS> -L 6006:localhost:6006
```

6.2. In a browser, visit .

6.3. In the top navigation bar, switch from `Graphs` to `Neuron`. If it does not show up, please wait a while and refresh the page while the plugin loads. If the issue persists, check the `Inactive` dropdown list on the right and check for `Neuron`.

6.4. If TensorBoard failed to find the generated logs, you will see the following message:

In this case, please check the console output on the Deployment Instance where TensorBoard was launched for any warnings or error messages, and make sure the version of the `aws-neuron-tools` package is compatible.

Visualize graphs executed on Neuron

Show how the graph was partition to run on NeuronCores

To view how the graph was partitioned to run on NeuronCores, select "Device" under "Graph Color Schemes" in the left navigation bar.

Each operator will be colored according to the device used. In this example, light blue indicates an operator was executed on CPU, and orange indicates the operator was executed on NeuronCores. Operators that are white may have been optimized by the Neuron compiler and fused into another operation.

Inspect which operators consumes the most time

You can also view how long each operator took by changing to the "Compute time" color scheme.

This view will show time taken by each layer and will be colored according to how much relative time the layer took to compute. A lighter shade of red means that a relatively small portion of compute time was spent in this layer, while a darker red shows that more compute time was used.

Check out Neuron support operators for each framework

The "Compatibility" color scheme allows you to better understand what operators are currently supported by the Neuron compiler - green for compatible ops, red for incompatible ops, and yellow for subgraphs that contain both compatible and incompatible ops.

Filter view by device

Additionally, you can choose to filter by CPU and NeuronCores, which will only color ops that match the selected device(s).

Expand/collapse subgraphs and view operator details

Each rectangular node in the graph represents a subgraph that can be expanded or collapse by clicking on the name. Operators will be represented by ellipses, and can be clicked to reveal more information on that operator, such as inputs and execution device.

The `Expand All` and `Collapse All` buttons can be used to expand or collapse every subgraph. When using these features, the positioning of the graph may change when redrawing the new graph. Try using `Reset Position` button and zoom out by scrolling if the graph appears to be missing.

Viewing the Neuron profile data

On the right side of the Neuron plugin, information on the profiled inference will be displayed.

See performance summary

First is the "Neuron Performance Summary," which gives a quick overview on how Neuron executed the graph, including information on the number of NeuronCores and both on-NeuronCore time and on-CPU time.

Get a breakdown of time spent per NeuronCore

Next, the "Neuron Execution" will give more details on how a graph was partitioned for Neuron. Each entry in the table will show the order it was executed in, what type of device was used, the compute time (in microseconds), and the percentage of total time spent. To dive deeper into subgraphs, you can check the "Show Details" box to display the breakdown per NeuronCore.

Get a breakdown of time spent per operator

The "Op Time Table" section shows the cycle count per operator, much like the "Compute time" coloring for graph visualization. This table can be sorted by clicking the column names, and searched using the provided text box in the top right corner. Due to Neuron compiler optimizations, some of the compute may not be associated with any specific operator and will be categorized as `unknown`. Additionally, time spent moving data to and from NeuronCores will fall under `(ND_ENGINE_LOAD)`.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

TensorBoard Plugin for Neuron

- *Neuron Plugin for TensorBoard (Trn1)*
- *Neuron Plugin for TensorBoard (Inf1)*
- *What's New*

This document is relevant for: `Inf1`, `Trn1`

13.3 Helper Tools

This document is relevant for: `Inf1`

13.3.1 Neuron Check Model

Overview

Neuron Check Model tool provides user with basic information about the compiled and uncompiled model's operations without the use of TensorBoard-Neuron. For additional visibility into the models, please see *Neuron Plugin for TensorBoard (Inf1)*.

Neuron Check Model tool scans the user's uncompiled model and provides a table of the operations within the uncompiled model. By default, the table shows each operation type and number of instances of that type within model, and whether the type is supported in Neuron. If –show_names option is specified, the table shows each operation by name and whether the type of that operation is supported in Neuron.

If the model is already compiled, the tool also provides the table of operations as for uncompiled model. The table include the Neuron subgraph type and number of instances of that type, along with operations that have not been compiled to Neuron. Additionally, the tool displays a message showing the minimum number of NeuronCores required to run the model, followed by another table which shows the list of Neuron subgraphs by name and the number of pipelined NeuronCores used by each subgraph. More information about NeuronCore pipeline can be found in *NeuronCore Pipeline*. If –expand_subgraph option is specified, the operations within each subgraph are printed below the subgraph information.

Neuron Check Model tool is currently available for TensorFlow and MXNet. To check PT model, please use torch.neuron.analyze_model function as shown in PyTorch-Neuron Getting Started tutorial *ResNet50 model for Inferentia*

TensorFlow-Neuron Check Model

The following example shows how to run TensorFlow-Neuron Check Model tool with TensorFlow ResNet50 tutorial.

1. Start with the TensorFlow ResNet50 tutorial at *Running ResNet50 on Inferentia* and do the first three steps of the tutorial. Please stay in the Python environment that you setup during the tutorial.

2. Install needed tensorflow_hub package and download the tool:

```
pip install tensorflow_hub
wget https://raw.githubusercontent.com/aws/aws-neuron-sdk/master/src/neuron-gatherinfo/
↪tf_neuron_check_model.py
python tf_neuron_check_model.py -h
```

```
usage: tf_neuron_check_model.py [-h] [--show_names] [--expand_subgraph]
                                model_path

positional arguments:
  model_path         a TensorFlow SavedModel directory (currently supporting
                     TensorFlow v1 SaveModel only).

optional arguments:
  -h, --help         show this help message and exit
  --show_names       list operation by name instead of summarizing by type
                     (caution: this option will generate many lines of output
                     for a large model).
  --expand_subgraph  show subgraph operations.
```

3. After step 3 of the TensorFlow ResNet50 tutorial, you can check the uncompiled model to see Neuron supported operations (currently supporting TensorFlow v1 SaveModel only):

```
$ python tf_neuron_check_model.py ws_resnet50/resnet50/

* The following table shows the supported and unsupported operations within this␣
↪uncompiled model.
* Each line shows an operation type, the number of instances of that type within model,
* and whether the type is supported in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['Placeholder', 'PlaceholderWithDefault', 'NoOp', 'Const', 'Identity', 'IdentityN',
↪'VarHandleOp',
 'VarIsInitializedOp', 'AssignVariableOp', 'ReadVariableOp', 'StringJoin',
↪'ShardedFilename', 'SaveV2',
 'MergeV2Checkpoints', 'RestoreV2']

Op Type           Num Instances   Neuron Supported ?
-------           -------------   ------------------
Pad               2               Yes
RandomUniform     54              Yes
Sub               54              Yes
Mul               54              Yes
Add               54              Yes
Conv2D            53              Yes
BiasAdd           54              Yes
FusedBatchNormV3  53              Yes
Relu              49              Yes
MaxPool           1               Yes
AddV2             16              Yes
Fill              56              Yes
Mean              1               Yes
MatMul            1               Yes
Softmax           1               Yes
Pack              1               Yes

* Total inference operations: 504
* Total Neuron supported inference operations: 504
* Percent of total inference operations supported by Neuron: 100.0
```

4. You can also check the compiled model to see the number of pipeline NeuronCores for each subgraph:

```
$ python tf_neuron_check_model.py ws_resnet50/resnet50_neuron/

* Found 1 Neuron subgraph(s) (NeuronOp(s)) in this compiled model.
* Use this tool on the original uncompiled model to see Neuron supported operations.
* The following table shows all operations, including Neuron subgraphs.
* Each line shows an operation type, the number of instances of that type within model,
* and whether the type is supported in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['Placeholder', 'PlaceholderWithDefault', 'NoOp', 'Const', 'Identity', 'IdentityN',
↪'VarHandleOp',
 'VarIsInitializedOp', 'AssignVariableOp', 'ReadVariableOp', 'StringJoin',
↪'ShardedFilename', 'SaveV2',
 'MergeV2Checkpoints', 'RestoreV2']

Op Type   Num Instances   Neuron Supported ?
-------   -------------   ------------------
NeuronOp  1               Yes

* Please run this model on Inf1 instance with at least 1 NeuronCore(s).
* The following list show each Neuron subgraph with number of pipelined NeuronCores used␣
↪by subgraph
* (and subgraph operations if --expand_subgraph is used):

Subgraph Name                                                                  Num␣
↪Pipelined NeuronCores
-------------                                                                  -----------
↪--------------
conv5_block3_3_bn/FusedBatchNormV3/ReadVariableOp/neuron_op_d6f098c01c780733  1
```

5. When showing subgraph information, you can use –expand_subgraph to show operation types in each subgraph:

```
$ python tf_neuron_check_model.py ws_resnet50/resnet50_neuron/ --expand_subgraph

(output truncated to show subgraph information only)

Subgraph Name                                                                  Num␣
↪Pipelined NeuronCores
-------------                                                                  -----------
↪--------------
conv5_block3_3_bn/FusedBatchNormV3/ReadVariableOp/neuron_op_d6f098c01c780733  1
     Op Type         Num Instances
     -------         -------------
     MatMul          1
     Relu            49
     Add             16
     FusedBatchNorm  53
     BiasAdd         54
     Conv2D          53
     Pad             2
     Mean            1
     MaxPool         1
```

(continues on next page)

(continued from previous page)

```
    Softmax         1
```

6. Use –show_names to see full operation names (caution: this option will generate many lines of output for a large model):

```
$ python tf_neuron_check_model.py ws_resnet50/resnet50_neuron/ --show_names

* Found 1 Neuron subgraph(s) (NeuronOp(s)) in this compiled model.
* Use this tool on the original uncompiled model to see Neuron supported operations.
* The following table shows all operations, including Neuron subgraphs.
* Each line shows an operation name and whether the type of that operation is supported␣
↪in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['Placeholder', 'PlaceholderWithDefault', 'NoOp', 'Const', 'Identity', 'IdentityN',
↪'VarHandleOp',
 'VarIsInitializedOp', 'AssignVariableOp', 'ReadVariableOp', 'StringJoin',
↪'ShardedFilename', 'SaveV2',
 'MergeV2Checkpoints', 'RestoreV2']

Op Name                                                                  Op Type   ␣
↪Neuron Supported ?
-------                                                                  -------    -
↪-----------------
conv5_block3_3_bn/FusedBatchNormV3/ReadVariableOp/neuron_op_d6f098c01c780733  NeuronOp ␣
↪Yes

* Please run this model on Inf1 instance with at least 1 NeuronCore(s).
* The following list show each Neuron subgraph with number of pipelined NeuronCores used␣
↪by subgraph
* (and subgraph operations if --expand_subgraph is used):

Subgraph Name                                                            Num␣
↪Pipelined NeuronCores
-------------                                                            -----------
↪---------------
conv5_block3_3_bn/FusedBatchNormV3/ReadVariableOp/neuron_op_d6f098c01c780733  1
```

MXNet-Neuron Check Model

The following example shows how to run MXNet-Neuron Check Model tool with MXNet ResNet50 tutorial.

1. Start with the MXNet ResNet50 tutorial at *Running Neuron Apache MXNet (Incubating) ResNet50 on Inferentia* and do the first three steps of the tutorial. Please stay in the Python environment that you setup during the tutorial.

2. Download the tool:

```
wget https://raw.githubusercontent.com/aws/aws-neuron-sdk/master/src/neuron-gatherinfo/
↪mx_neuron_check_model.py
python mx_neuron_check_model.py -h
```

```
usage: mx_neuron_check_model.py [-h] [--show_names] [--expand_subgraph]
                                model_path

positional arguments:
  model_path         path prefix to MXNet model (the part before -symbol.json)

optional arguments:
  -h, --help         show this help message and exit
  --show_names       list operation by name instead of summarizing by type
                     (caution: this option will generate many lines of output
                     for a large model).
  --expand_subgraph  show subgraph operations.
```

3. After step 3 of MXNet ResNet50 tutorial, you can check the uncompiled model to see Neuron supported operations:

```
$ python mx_neuron_check_model.py resnet-50

* The following table shows the supported and unsupported operations within this␣
↪uncompiled model.
* Each line shows an operation type, the number of instances of that type within model,
* and whether the type is supported in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['null']

Op Type         Num Instances   Neuron Supported ?
-------         -------------   ------------------
BatchNorm       51              Yes
Convolution     53              Yes
Activation      50              Yes
Pooling         2               Yes
elemwise_add    16              Yes
Flatten         1               Yes
FullyConnected  1               Yes
SoftmaxOutput   1               No

* Total inference operations: 175
* Total Neuron supported inference operations: 174
* Percent of total inference operations supported by Neuron: 99.4
```

4. You can also check the compiled model to see the number of pipeline NeuronCores for each subgraph:

```
$ python mx_neuron_check_model.py resnet-50_compiled

* Found 1 Neuron subgraph(s) (_neuron_subgraph_op(s)) in this compiled model.
* Use this tool on the original uncompiled model to see Neuron supported operations.
* The following table shows all operations, including Neuron subgraphs.
* Each line shows an operation type, the number of instances of that type within model,
* and whether the type is supported in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['null']
```

(continues on next page)

(continued from previous page)

```
Op Type               Num Instances   Neuron Supported ?
-------               -------------   ------------------
_neuron_subgraph_op   1               Yes
SoftmaxOutput         1               No

* Please run this model on Inf1 instance with at least 1 NeuronCore(s).
* The following list show each Neuron subgraph with number of pipelined NeuronCores used␣
↪by subgraph
* (and subgraph operations if --expand_subgraph is used):

Subgraph Name         Num Pipelined NeuronCores
-------------         -------------------------
_neuron_subgraph_op0  1
```

5. When showing subgraph information, you can use –expand_subgraph to show operation types in each subgraph:

```
$ python mx_neuron_check_model.py resnet-50_compiled --expand_subgraph

(output truncated to show subgraph information only)

Subgraph Name         Num Pipelined NeuronCores
-------------         -------------------------
_neuron_subgraph_op0  1
     Op Type         Num Instances
     -------         -------------
     BatchNorm       51
     Convolution     53
     Activation      50
     Pooling         2
     elemwise_add    16
     Flatten         1
     FullyConnected  1
```

6. Use –show_names to see full operation names (caution: this option will generate many lines of output for a large model):

```
$ python mx_neuron_check_model.py resnet-50_compiled --show_names

* Found 1 Neuron subgraph(s) (_neuron_subgraph_op(s)) in this compiled model.
* Use this tool on the original uncompiled model to see Neuron supported operations.
* The following table shows all operations, including Neuron subgraphs.
* Each line shows an operation name and whether the type of that operation is supported␣
↪in Neuron.
* Some operation types are excluded from table because they are no-operations or␣
↪training-related operations:
 ['null']

Op Name               Op Type              Neuron Supported ?
-------               -------              ------------------
_neuron_subgraph_op0  _neuron_subgraph_op  Yes
softmax               SoftmaxOutput        No
```

(continues on next page)

(continued from previous page)

```
* Please run this model on Inf1 instance with at least 1 NeuronCore(s).
* The following list show each Neuron subgraph with number of pipelined NeuronCores used␣
↪by subgraph
* (and subgraph operations if --expand_subgraph is used):

Subgraph Name         Num Pipelined NeuronCores
-------------         -------------------------
_neuron_subgraph_op0  1
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

13.3.2 Using Neuron GatherInfo Tool to collect debug and support information

Overview

The Neuron GatherInfo tool `neuron-gatherinfo.py` can assist in automating the collection and packaging of information from Neuron SDK tools that is useful to both user and AWS for issue resolution. The tool gathers log files and other system information. If being used to supply that info to AWS, the tool will redact proprietary and confidential information. The GatherInfo tool is supplied in source code form - available here: Neuron Gatherinfo

The tool enables developers to gather compiler and inference/runtime logs. Additionally, the common usage is from within one of the supported ML frameworks that have been integrated with Neuron, and information can be captured from those compile/runtime environments using the frameworks.

Steps Overview:

1. Obtain a copy of neuron-gatherinfo.py from Neuron Gatherinfo
2. Install into a location in your $PATH or into a location from where you can launch the script
3. Use with compile and/or runtime environments

Neuron-CC information gathering

Step 1: Re-run the compile steps for your workload with increased verbosity or debug levels

- For TensorFlow-Neuron, change the Python code as shown. Note that 'compiler-workdir' is expected to be an empty directory to prevent files from other runs from interfering with the information gathering. The call to the compile function has to be augmented with the **verbose** and the **compiler_workdir **arguments. In addition, please capture the stdout messages into a file (for example, by redirecting the stdout to a file)

```
tfn.saved_model.compile(model_dir, compiled_model_dir, compiler_args=['--verbose', '2',
↪'--pipeline', 'compile',  'SaveTemps'], compiler_workdir='./compiler-workdir')
```

- For Neuron Apache MXNet (Incubating), add compiler arguments as shown below and run the compilation process from an empty workdir:

```
import mxnet as mx
import os

from packaging import version
mxnet_version = version.parse(mx.__version__)
if mxnet_version >= version.parse("1.8"):
   import mx_neuron as neuron
else:
   from mxnet.contrib import neuron

...
os.environ['SUBGRAPH_INFO'] = '1'
compile_args = { '--verbose' : 2, '--pipeline' : 'compile', 'flags' : ['SaveTemps'] }
csym, cargs, cauxs = neuron.compile(sym, args, auxs, inputs=inputs, **compile_args)
```

Step 2: Run neuron-gatherinfo.py to gather information to share

The output result will be a tar.gz file.

Neuron Runtime information gathering

Step 1: EXECUTE inference steps for your workload with increased verbosity or debug levels

In the case of runtime information, the tool **neuron-dump.py** is used by **neuron-gatherinfo.py **to gather that information. Make sure that you have the neuron tools package (aws-neuron-tools) installed.

Step 2: Run neuron-gatherinfo.py to gather information to share

The output result will be a tar.gz file.

Tool Usage Reference

Run neuron-gatherinfo.py using the "—help" option:

```
bash $ ~/bin/neuron-gatherinfo.py --help
usage: neuron-gatherinfo.py [-h] [--additionalfileordir ADDFLDIR] [-c CCDIR]
                            [-i] [-f FILTERFILE] [-m] -o OUTDIR [-r RTDIR] -s
                            STDOUT [-v]

    Usage: /home/user/bin/neuron-gatherinfo.py [options]
    This program is used to gather information from this system for analysis
    and debugging


optional arguments:
  -h, --help            show this help message and exit
  --additionalfileordir ADDFLDIR
                        Additional file or directory that the user wants to
```

(continues on next page)

(continued from previous page)

```
                        provide in the archive. The user can sanitize this
                        file or directory before sharing
  -c CCDIR, --compileroutdir CCDIR
                        Location of the neuron-cc generated files
  -i, --include         By default, only the lines containing (grep) patterns
                        like 'nrtd|neuron|kernel:' from the syslog are copied.
                        Other lines are excluded. Using this option allows the
                        timestamp section of other lines to be included. The
                        rest of the contents of the line itself are elided.
                        Providing the timestamp section may provide time
                        continuity while viewing the copied syslog file
  -f FILTERFILE, --filter FILTERFILE
  -m, --modeldata       By using this option, the entire compiler work
                        directory's contents will be included (excluding the
                        .pb files, unless an additional option is used). This
                        would include model information, etc. The files that
                        are included, by default, are these: graph_def.neuron-
                        cc.log, all_metrics.csv, hh-tr-operand-
                        tensortensor.json
  -o OUTDIR, --out OUTDIR
                        The output directory where all the files and other
                        information will be stored. The output will be stored
                        as an archive as well as the actual directory where
                        all the contents are copied. This will allow a simple
                        audit of the files, if necessary. *** N O T E ***:
                        Make sure that this directory has enough space to hold
                        the files and resulting archive
  -r RTDIR, --runtimeoutdir RTDIR
                        Location of the neuron runtime generated files
  -s STDOUT, --stdout STDOUT
                        The file where the stdout of the compiler run was
                        saved
  -v, --verbose         Verbose mode displays commands executed and any
                        additional information which may be useful in
                        debugging the tool itself
```

Examples

Example 1: no ML model information gathered (default behavior)

In this case, the tool will archive just the default information gathering:

```
bash $ sudo ~/bin/neuron-gatherinfo.py   -o compile-and-run-info-for-debugging-no-model-
↪info  -i --verbose  -s stdout-from-compile_resnet50.out -c compiler-workdir

Running cmd: lscpu and capturing output in file: /home/user/tutorials-3/compile-and-run-
↪info-for-debugging-no-model-info/neuron-gatherinfo/report-lscpu.txt
Running cmd: lshw and capturing output in file: /home/user/tutorials-3/compile-and-run-
↪info-for-debugging-no-model-info/neuron-gatherinfo/report-lshw.txt
Running cmd: lspci | grep -i Amazon and capturing output in file: /home/user/tutorials-3/
↪compile-and-run-info-for-debugging-no-model-info/neuron-gatherinfo/report-lspci.txt
```

(continues on next page)

(continued from previous page)

```
Running cmd: neuron-cc --version and capturing output in file: /home/user/tutorials-3/
↪compile-and-run-info-for-debugging-no-model-info/neuron-gatherinfo/report-neuron-cc.txt
Running cmd: neuron-ls and capturing output in file: /home/user/tutorials-3/compile-and-
↪run-info-for-debugging-no-model-info/neuron-gatherinfo/report-neuron-ls.txt
<SNIP>
    ******
    Archive created at:
        /home/user/tutorials-3/compile-and-run-info-for-debugging-no-model-info/neuron-
↪gatherinfo.tar.gz
    From directory:
        /home/user/tutorials-3/compile-and-run-info-for-debugging-no-model-info/neuron-
↪gatherinfo
    ******
```

Example 2 : model ML information gathered using the "—modeldata" option

In this case, the tool will archive the compiler work directory in addition to the default information gathering

```
bash $ sudo ~/bin/neuron-gatherinfo.py   -o compile-and-run-info-for-debugging  -i --
↪verbose  -s stdout-from-compile_resnet50.out -c compiler-workdir --modeldata

<SNIP>
Running cmd: lscpu and capturing output in file: /home/user/tutorials-3/compile-and-run-
↪info-for-debugging/neuron-gatherinfo/report-lscpu.txt
Running cmd: lshw and capturing output in file: /home/user/tutorials-3/compile-and-run-
↪info-for-debugging/neuron-gatherinfo/report-lshw.txt
Running cmd: lspci | grep -i Amazon and capturing output in file: /home/user/tutorials-3/
↪compile-and-run-info-for-debugging/neuron-gatherinfo/report-lspci.txt
Running cmd: neuron-cc --version and capturing output in file: /home/user/tutorials-3/
↪compile-and-run-info-for-debugging-no-model-info/neuron-gatherinfo/report-neuron-cc.txt
Running cmd: neuron-ls and capturing output in file: /home/user/tutorials-3/compile-and-
↪run-info-for-debugging-no-model-info/neuron-gatherinfo/report-neuron-ls.txt
<SNIP>

    ******
    Archive created at:
        /home/user/tutorials-3/compile-and-run-info-for-debugging/neuron-gatherinfo.tar.
↪gz
    From directory:
        /home/user/tutorials-3/compile-and-run-info-for-debugging/neuron-gatherinfo
    ******


    **************************
    Based on your command line option, we're also packaging these files:

        graph_def.neuron-cc.log
        all_metrics.csv
        hh-tr-operand-tensortensor.json

    And this directory: /home/user/tutorials-3/compiler-workdir
```

(continues on next page)

(continued from previous page)

```
**************************
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

Helper Tools

- *Neuron Check Model*
- *Using Neuron GatherInfo Tool to collect debug and support information*

This document is relevant for: `Inf1`

13.4 NeuronPerf (Beta)

NeuronPerf is a lightweight Python library with a simple API that enables fast measurements of performance when running models using Neuron.

13.4.1 NeuronPerf Quickstart

To install NeuronPerf in your Neuron environment, execute:

```
$ pip install neuronperf --extra-index-url=https://pip.repos.neuron.amazonaws.com
```

Refer to the *NeuronPerf Examples* and *NeuronPerf User Guide* to get started.

13.4.2 NeuronPerf User Guide

This document is relevant for: `Inf1`

NeuronPerf Overview

NeuronPerf is a lightweight Python library that can help you easily benchmark your models with Neuron hardware.

NeuronPerf supports Neuron releases for PyTorch, Tensorflow, and MXNet. It is used internally by the Neuron team to generate performance benchmarking numbers.

When interacting with NeuronPerf, you will typically import the base package along with one of the submodule wrappers, for example:

```
import neuronperf
import neuronperf.torch
```

You may then benchmark and/or compile one or more models with NeuronPerf. For example,

```
reports = neuronperf.torch.benchmark(model, inputs, ...)
```

The `compile` and `benchmark` methods must be accessed through one of the supported framework submodules.

Benchmarking

All NeuronPerf `benchmark` calls require a minimum of two arguments:

1. A filename
2. Inputs

The filename may refer to:

1. A Neuron-compiled model (e.g. `my_model.pt`)
2. A *Model Index*.

A Model Index is useful for benchmarking more than one model in a single session.

Compiling

NeuronPerf also provides a standard interface to all Neuron frameworks through the `compile` API.

```
model_index = neuronperf.torch.compile(model, inputs, ...)
```

This is completely optional. You may use the standard compilation guides for supported frameworks.

Next Steps

Take a look at the simple *NeuronPerf Examples*, *NeuronPerf Benchmark Guide*, *NeuronPerf Compile Guide*, and *NeuronPerf API*.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Terminology

- Model Inputs - An individual input or `list` of inputs - Example: `inputs = [(torch.ones((batch_size, 5))) for batch_size in batch_sizes]` - Each input is associated with the `batch_sizes` specified, in the same order - Each input is fed individually to a corresponding model - If an input is provided as a `tuple`, it will be destructured to `model(*input)` to support multiple args - See *NeuronPerf Framework Notes* for framework-specific requirements
- **Latency**
  - Time to execute a single `model(input)`
  - Typically measured in milliseconds
- **Model**
  - Your data model; varies by framework. See *NeuronPerf Framework Notes*
  - Models may be wrapped by submodules (`torch`, `tensorflow`, `mxnet`) as callables
- **Model Index**
  - A JSON file that tracks compiled model artifacts
- **Model Inputs**
  - A `tuple` of inputs passed to a model, i.e. a single complete example

  - Example: `input = (torch.ones((5, 3, 224, 224)),)`

- **Throughput**
  - Inferences / second

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Examples

This page walks through several examples of using NeuronPerf, starting with the simplest way—using a compiled model. We will also see how we can use NeuronPerf to perform a hyperparameter search, and manage the artifacts produced, as well as our results.

Benchmark a Compiled Model

This example assumes you have already compiled your model for Neuron and saved it to disk. You will need to adapt the batch size, input shape, and filename for your model.

```
import torch  # or tensorflow, mxnet

import neuronperf as npf
import neuronperf.torch  # or tensorflow, mxnet

# Construct dummy inputs
batch_sizes = 1
input_shape = (batch_sizes, 3, 224, 224)
inputs = torch.ones(input_shape)  # or numpy array for TF, MX

# Benchmark and save results
reports = npf.torch.benchmark("your_model_file.pt", inputs, batch_sizes)
npf.print_reports(reports)
npf.write_csv(reports)  # save table of most useful data, or...
npf.write_json(reports)  # save everything
```

```
INFO:neuronperf.benchmarking - Benchmarking 'your_model_file.pt', ~8.0 minutes remaining.
throughput_avg    latency_ms_p50    latency_ms_p99    n_models          pipeline_size   ␣
↪ workers_per_model batch_size        model_filename
296766.5          0.003             0.003             1                 1               ␣
↪ 1                 1                 your_model_file.pt
3616109.75        0.005             0.008             24                1               ␣
↪ 1                 1                 your_model_file.pt
56801.0           0.035             0.04              1                 1               ␣
↪ 2                 1                 your_model_file.pt
3094419.4         0.005             0.051             24                1               ␣
↪ 2                 1                 your_model_file.pt
```

Let's suppose you only wish to test two specific configurations. You wish to benchmark 1 model and 1 worker thread, and also with 2 worker threads for 15 seconds each. The call to `benchmark` becomes:

```
reports = npf.torch.benchmark(filename, inputs, batch_sizes, n_models=1, workers_per_
↪model=[1, 2], duration=15)
```

You can also add a custom model name to reports.

```
reports = npf.torch.benchmark(..., model_name="MyFancyModel")
```

See the *NeuronPerf Benchmark Guide* for further details.

Benchmark a Model from Source

In this example, we define, compile, and benchmark a simple (dummy) model using PyTorch.

We'll assume you already have a PyTorch model compiled for Neuron with the filename `model_neuron_b1.pt`. Furthermore, let's assume the model was traced with a batch size of 1, and has an input shape of (3, 224, 224).

Listing 1: `test_simple_pt.py`

```
import torch
import torch.neuron

import neuronperf as npf
import neuronperf.torch


# Define a simple model
class Model(torch.nn.Module):
    def forward(self, x):
        x = x * 3
        return x + 1


# Instantiate
model = Model()
model.eval()

# Define some inputs
batch_sizes = [1]
inputs = [torch.ones((batch_size, 3, 224, 224)) for batch_size in batch_sizes]

# Compile for Neuron
model_neuron = torch.neuron.trace(model, inputs)
model_neuron.save("model_neuron_b1.pt")

# Benchmark
reports = npf.torch.benchmark("model_neuron_b1.pt", inputs, batch_sizes)

# View and save results
npf.print_reports(reports)
npf.write_csv(reports, "model_neuron_b1.csv")
```

```
(aws_neuron_pytorch_p36) ubuntu@ip-172-31-11-122:~/tmp$ python test_simple_pt.py
INFO:neuronperf.benchmarking - Benchmarking 'model_neuron_b1.pt', ~8.0 minutes remaining.
throughput_avg    latency_ms_p50    latency_ms_p99    n_models          pipeline_size
↪ workers_per_model batch_size        model_filename
296766.5          0.003             0.003             1                 1
↪ 1                 1                 model_neuron_b1.pt
```

(continues on next page)

(continued from previous page)

```
3616109.75          0.005              0.008              24                 1                 ␣
↪ 1                 1                  model_neuron_b1.pt
56801.0             0.035              0.04               1                  1                 ␣
↪ 2                 1                  model_neuron_b1.pt
3094419.4           0.005              0.051              24                 1                 ␣
↪ 2                 1                  model_neuron_b1.pt
```

Great! Here is what a default `csv` file looks like.

n_models	workers_per_model	batch_size	num_cores	pipeline_cores	latency_p0	latency_p25	latency_p50	latency_p90	latency_p95	latency_p99	latency_p100	avg_throughput	avg_latency_ms	pipeline_throughput	pipeline_latency_ms	avg_model_load_time_ms	worker_avg_inference_time_ms	worker_avg_throughput_inf/s	total_inferences	total_time_s	model_status	multi_process	multi_interpreter	client_type	instance_type
1	1	1	1	3134961.0	0.008	0.03	0.03	0.03	0.03	0.03	7.73	62.21	17.62	5.03	0.00	1.0	0.0	0.02	48.93	154704	finished	True	False	neuron-bench	inf1.6xlarge
16	16	1	1	3806084.0	0.032	0.03	0.05	0.05	0.05	0.05	7.93	293.3	80.26	6.04	0.00	1.0	0.0	0.03	49.7	1799549	finished	True	False	neuron-bench	inf1.6xlarge
1	2	1	1	5117581.0	0.036	0.03	0.03	0.03	0.04	7.13	114.2	17.8	10.03	7.00	0.00	1.0	0.0	0.03	43.88	248984	finished	True	False	neuron-bench	inf1.6xlarge
16	32	1	1	3810988.0	0.05	0.06	0.07	0.07	0.12	48.0	703.4	91.2	20.08	0.00	0.00	1.0	0.0	0.07	44.69	180490	finished	True	False	neuron-bench	inf1.6xlarge

Compile and Benchmark a Model

Here is an end-to-end example of compiling and benchmarking a ResNet-50 model from `torchvision`.

Listing 2: `test_resnet50_pt.py`

```
import torch
import torch_neuron

import neuronperf as npf
import neuronperf.torch

from torchvision import models


# Load a pretrained ResNet50 model
model = models.resnet50(pretrained=True)

# Select a few batch sizes to test
filename = 'resnet50.json'
batch_sizes = [5, 6, 7]

# Construct example inputs
inputs = [torch.zeros([batch_size, 3, 224, 224], dtype=torch.float32) for batch_size in␣
↪batch_sizes]

# Compile
npf.torch.compile(
```

(continues on next page)

(continued from previous page)

```
        model,
        inputs,
        batch_sizes=batch_sizes,
        filename=filename,
)

# Benchmark
reports = npf.torch.benchmark(filename, inputs)

# View and save results
npf.print_reports(reports)
npf.write_csv(reports, 'resnet50_results.csv')
npf.write_json(reports, 'resnet50_results.json')
```

Benchmark on CPU or GPU

When benchmarking on CPU or GPU, the API is slightly different. With CPU or GPU, there is no compiled model to benchmark, so instead we need to directly pass a reference to the model class that will be instantiated.

Note: GPU benchmarking is currently only available for PyTorch.

CPU:

```
cpu_reports = npf.cpu.benchmark(YourModelClass, ...)
```

GPU:

```
gpu_reports = npf.torch.benchmark(YourModelClass, ..., device_type="gpu")
```

Please refer to *Benchmark on CPU or GPU* for details and an example of providing your model class.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Benchmark Guide

The call to `neuronperf[torch/tensorflow/mxnet/cpu].benchmark` is used to measure your model performance. It will choose reasonable defaults if none are provided, and will return back reports that summarize the benchmarking results.

What is the default behavior of `benchmark`?

That will depend how you provided your model and how your model was compiled.

The two most common ways to provide your model are:

1. Provide the path to your compiled model
2. Provide the path to a model index from `neuronperf.compile` (a JSON file)

Data Parallel

If you choose 1, then…

Your model is benchmarked on provided `inputs` in 4 different configurations:

1. A single model on 1 NeuronCore with one worker (min. latency)
2. A single model on 1 NeuronCore with two workers (max. throughput / NC)
3. `MAX` models on `MAX` NeuronCores with one worker (min. latency + max. instance usage)
4. `MAX` models on `MAX` NeuronCores with two workers (max. throughput + max. instance usage)

The value `MAX` is automatically determined by your instance size. If it can't be identified, those configurations will be skipped.

The primary benefit of (3) and (4) is to verify that your model scales well at maximum instance usage.

If you chose 2, then…

- Your input parameters to `benchmark` (`batch_sizes`, etc.) are treated as filters on the index
- Each remaining model configuration is benchmarked as described in (1)

Pipeline

Pipeline mode is active when using a Neuron device and `pipeline_sizes > 1`. The same behavior as described in Data Parallel applies, except that only one worker configuration is executed: the optimal number of workers for your pipeline size, unless manually overridden.

Parameters

Below are some useful and common parameters to tweak. Please see the *NeuronPerf API* for full details.

- `n_models` controls how many models to load. The default behavior is `n_models=[1, MAX]`.
- `workers_per_model` controls how many worker threads will be feeding inputs to each model. The default is automatically determined.
- `pipeline_sizes` tells the benchmarker how many cores are needed for your model so that each model instance can be loaded properly. Default is 1.
- `duration` controls how long to run each configuration.
- `batch_sizes` is used to inform the benchmarker of your input shape so that throughput can be computed correctly.

Almost all NeuronPerf behaviors are controllable via arguments found in the *NeuronPerf API*. This guide attempts to provide some context and examples for those arguments.

Inputs

Models accept one or more inputs to operate on. Since NeuronPerf needs to support multiple inputs for multiple models, as well as multi-input models, there are some details that may need your attention. See the *NeuronPerf Framework Notes* for details.

Multi-input Models

If your model accepts multiple inputs, you must provide them in a `tuple`. For example, suppose you have a model like this:

```
class Model(torch.nn.Module):
        def forward(self, x, y, z):
                ...
                return output
```

In order for NeuronPerf to pass along your multiple inputs correctly, you should provide them as a `tuple`:

```
inputs = (x, y, z)
npf.torch.benchmark(model_filename, inputs, ...)
```

If you are compiling and/or benchmarking multiple models, you can pass different sized inputs as a list of tuples:

```
inputs = [(x1, y1, z1), (x2, y2, z2), ...]
npf.torch.benchmark(model_filename, inputs, ...)
```

Preprocessing and Postprocessing

Many models have additional preprocessing and postprocessing steps involved that may add non-negligible overhead to inference time. NeuronPerf supports these use cases through the use of custom functions.

Preprocessing

Recall that NeuronPerf expects (or wraps) each model input into a `tuple`. These tuples will be unpacked before calling your model.

Here is an example for a model with one input. The example multiples the input by 5 before inference.

```
def preprocess_fn(x):
    return x * 5

...

# Benchmark with custom preprocessing function
reports = npf.torch.benchmark(
        filename,
        inputs,
        ...,
        preprocess_fn = preprocess_fn,
)
```

Or if your model expects multiple inputs:

```
def preprocess_fn(x, y, z):
    return x / 255, y / 255, z / 255

...

# Benchmark with custom preprocessing function
reports = npf.torch.benchmark(
        filename,
        inputs,
        ...,
        preprocess_fn = preprocess_fn,
)
```

Postprocessing

Postprocessing is almost identical to preprocessing, except that your function will receive whatever the output of your model is, exactly as returned without modification. There are no type guarantees.

```
def postprocess_fn(x):
   return x.argmax()

...

# Benchmark with custom preprocessing function
reports = npf.torch.benchmark(
      filename,
      inputs,
      ...,
      postprocess_fn = postprocess_fn,
)
```

Minimal Latency

Suppose you are interested in the minimal latency achievable with your model. In this case, there is no need for more than one worker to execute at a time. We can manually specify the number of workers to use. See below *Worker Threads*.

Worker Threads

The argument `workers_per_model` controls the number of worker threads that are trying to prepare and load examples onto a single NeuronCore at a time. Therefore, a value of 1 corresponds to 1 thread / model. If `n_models=16`, then there would be 16 worker threads, one per model. This number is selected based upon whether you are using DataParallel (i.e. `pipeline_sizes == 1`), or Pipeline Mode (`pipeline_sizes != 1`).

By default, NeuronPerf will try to pick try multiple combinations of model copies and workers. You may be interested in controlling this manually.

```
reports = npf.torch.benchmark('model_neuron_b1.pt', ..., workers_per_model=1)
```

You may also pass a list, as with other parameters:

```
workers_per_model = [1, 2] # Same as the default for data parallel
reports = npf.torch.benchmark('model_neuron_b1.pt', ..., workers_per_
↪model)
```

With the default number of *Model Copies*, a call to `print_results` might look like this:

```
throughput_avg latency_ms_p50 latency_ms_p99 n_models       pipeline_size  workers_per_
↪model batch_size     model_filename
307.25         3.251          3.277          1              1              1            ␣
↪    1              models/a5cff386-89ca-4bbf-9087-d0e624c3c604.pt
2746.0         5.641          6.82           16             1              1            ␣
↪    1              models/a5cff386-89ca-4bbf-9087-d0e624c3c604.pt
329.5          6.053          6.108          1              1              2            ␣
↪    1              models/a5cff386-89ca-4bbf-9087-d0e624c3c604.pt
2809.0         10.246         12.52          16             1              2            ␣
↪    1              models/a5cff386-89ca-4bbf-9087-d0e624c3c604.pt
```

Model Copies

By default, NeuronPerf will benchmark two settings for `n_models`:

1. A single copy
2. The maximum number number of copies for your instance size

You can override this behavior by passing `n_models` to `benchmark`, as shown below:

```
reports = npf.torch.benchmark('model_neuron_b1.pt', ..., n_models=6)
```

or

```
n_models = list(range(1, 10))
reports = npf.torch.benchmark('model_neuron_b1.pt', ..., n_models=n_models)
```

Pipeline Mode

By default, NeuronPerf will assume you intend to use DataParallel, with two exceptions:

- You compiled your model using NeuronPerf for pipeline mode
- You constructed a neuronperf_model_index that uses pipeline mode

You can also manually tell NeuronPerf that your model was compiled for pipeline mode. It is similar to how other arguments are passed.

```
reports = npf.torch.benchmark('model_neuron_b1.pt', ..., pipeline_sizes=2)
```

If you are passing multiple models in an index, then you should pass a list for `pipeline_sizes`.

```
reports = npf.torch.benchmark('model_index.json', ..., pipeline_sizes=[1, 2, 3])
```

Duration

NeuronPerf will benchmark each configuration specified for 60 seconds by default. You can control the duration by passing `duration` (in seconds).

```
reports = npf.torch.benchmark('model_index.json', ..., duration=10)
```

> **Warning:** If you make the duration too short, it may expire before all models are loaded and have had time to execute.

Custom Datasets (Beta)

Currently, only PyTorch supports custom datasets, and the interface is subject to change. If you provide a custom dataset, it will be fully executed on each loaded model copy. So if you provide `n_models=2`, your dataset will be run through twice in parallel.

To use this API, call `benchmark` passing a `torch.utils.data.Dataset` to `inputs`. You can easily create your own `Dataset` by implementing the interface, or use one of the available datasets. For example:

```
import torchvision

dataset = torchvision.datasets.FashionMNIST(
   root="data",
   train=False,
   download=True,
   transform=ToTensor()
)

reports = npf.torch.benchmark('model_index.json', inputs=dataset, batch_sizes=[8],␣
↪preprocess_fn=lambda x: x[0], loop_dataset=False)
```

Note: The `preprocess_fn` is required here to extract image input from the `(image, label)` tuple generated by dataloader. If the length of dataset is not sufficient to get the runtime performance, one can set `loop_dataset=True` to rerun dataset until certain duration.

Results

Viewing and Saving

There are currently three ways to view results.

- **`neuronperf.print_reports(...)`**
  - Dump abbrieviated results in your terminal
- **`neuronperf.write_csv(...)`**
  - Store metrics of interest as CSV
- **`neuronperf.write_json(...)`**

- Store everything as JSON

See the *NeuronPerf API* for full details.

Full Timing Results

NeuronPerf automatically combines and summarizes the detailed timing information collecting during benchmarking. If you wish to receive everything back yourself, you can use:

```
results = npf.torch.benchmark('model_index.json', ..., return_timers=True)
```

If you later wish to produce reports the same way that NeuronPerf does internally, you can call:

```
reports = npf.get_reports(results)
```

Verbosity

Verbosity is an integer, currently one of `{0, 1, 2}`, where:

- 0 = SILENT
- 1 = INFO (default)
- 2 = VERBOSE / DEBUG

Example:

```
reports = npf.torch.benchmark(..., n_models=1, duration=5, verbosity=2)
```

```
DEBUG:neuronperf.benchmarking - Cast mode was not specified, assuming default.
INFO:neuronperf.benchmarking - Benchmarking 'resnet50.json', ~5 seconds remaining.
DEBUG:neuronperf.benchmarking - Running model config: {'model_filename': 'models/model_
↪b1_p1_83bh3hhs.pt', 'device_type': 'neuron', 'input_idx': 0, 'batch_size': 1, 'n_models
↪': 1, 'workers_per_model': 2, 'pipeline_size': 1, 'cast_mode': None, 'multiprocess':␣
↪True, 'multiinterpreter': False, 'start_dts': '20211111-062818', 'duration': '5'}
DEBUG:neuronperf.benchmarking - Benchmarker 0 started.
DEBUG:neuronperf.benchmarking - Benchmarker 0, Worker 0 started.
DEBUG:neuronperf.benchmarking - Benchmarker 0, Worker 1 started.
DEBUG:neuronperf.benchmarking - Benchmarker 0, Worker 0 finished after 738 inferences.
DEBUG:neuronperf.benchmarking - Benchmarker 0, Worker 1 finished after 738 inferences.
DEBUG:neuronperf.benchmarking - Benchmarker 0 finished.
throughput_avg latency_ms_p50 latency_ms_p99 n_models       pipeline_size  workers_per_
↪model batch_size     model_filename
329.667        6.073          6.109          1              1              2            ␣
↪     1              models/model_b1_p1_83bh3hhs.pt
```

Internal Process Model

For each model loaded (see *Model Copies*), a process is spawned. Each process may use multiple threads (see *Worker Threads*). The threads will continue to load examples and keep the hardware busy.

NeuronPerf spawns processes slightly differently between frameworks. For PyTorch and Apache MXNet (Incubating), processes are forked. For Tensorflow/Keras, a fresh interpreter is launched, and benchmarkers are serialized and run as a script.

If you suspect you are having trouble due to the way processes are managed, you have two mechanisms of control:

```
reports = npf.torch.benchmark(..., multiprocess=False)
```

Default is `True`, and `False` will disable multiprocessing and run everything inside a single parent process. This may not work for all frameworks beyond the first model configuration, because process teardown is used to safely deallocate models from the hardware. It is not recommeneded to benchmark this way.

```
reports = npf.torch.benchmark(..., multiinterpreter=True)
```

This flag controls whether a fresh interpreter is used instead of forking. Defaults to `False` except with Tensorflow/Keras.

Benchmark on CPU or GPU

When benchmarking on CPU or GPU, the API is slightly different. With CPU or GPU, there is no compiled model to benchmark, so instead we need to directly pass a reference to the model class that will be instantiated.

Note: GPU benchmarking is currently only available for PyTorch.

CPU:

```
cpu_reports = npf.cpu.benchmark(YourModelClass, ...)
```

GPU:

```
gpu_reports = npf.torch.benchmark(YourModelClass, ..., device_type="gpu")
```

Your model class will be instantiated in a subprocess, so there are some things to keep in mind.

- **Your model class must be defined at the top level inside a Python module**
  - i.e. don't place your model class definition inside a function or other nested scope
- If your model class has special Python module dependencies, consider importing them inside your class `__init__`
- If your model class expects constructor arguments, wrap your class so that it has no constructor arguments

Example of a wrapped model class for CPU/GPU benchmarking:

```
class ModelWrapper(torch.nn.Module):
   def __init__(self):
      super().__init__()
      from transformers import AutoModelForSequenceClassification
      model_name = "bert-base-cased"
```

(continues on next page)

(continued from previous page)

```
        self.bert = AutoModelForSequenceClassification.from_pretrained(model_name, return_
↪dict=False)
        self.add_module(model_name, self.bert)

    def forward(self, *inputs):
        return self.bert(*inputs)


reports = npf.torch.benchmark(ModelWrapper, inputs, device_type="gpu")
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Evaluate Guide

NeuronPerf has a new API for evaluating model accuracy on Neuron hardware. This API is currently only available for PyTorch.

You can access the API through standard `benchmark()` by passing an additional kwarg, `eval_metrics`.

For example:

```
reports = npf.torch.benchmark(
    model_index_or_path,
    dataset,
    n_models=1,
    workers_per_model=2,
    duration=0,
    eval_metrics=['accuracy', 'precision']
)
```

In this example, we fix `n_models` and `n_workers` because replicating the same model will not impact accuracy. We also set `duration=0` to allow benchmarking to run untimed through all dataset examples.

Because this call can be tedious to type, a convenience function is provided:

```
reports = npf.torch.evaluate(model_index_or_path, dataset, metrics=['accuracy',
↪'precision'])
```

The `dataset` can be any iterable object that produces `tuple(*INPUTS, TARGET)`.

If `TARGET` does not appear in the last column for your dataset, you can customize this by passing `eval_target_col`.

For example:

```
reports = npf.torch.evaluate(model_index_or_path, dataset, metrics='accuracy', eval_
↪target_col=1)
```

You can list the currently available metrics.

```
>>> npf.list_metrics()                                                          ␣
↪              |.....
Name                      Description                                           ␣
↪              |.....
```

(continues on next page)

(continued from previous page)

```
Accuracy                 (TP + TN) / (TP + TN + FP + FN)
↪              |.....
TruePositiveRate         TP / (TP + FN)
↪              |.....
Sensitivity              Alias for TruePositiveRate
↪              |.....
Recall                   Alias for TruePositiveRate
↪              |.....
Hit Rate                 Alias for TruePositiveRate
↪              |.....
TrueNegativeRate         TN / (TN + FP)
↪              |.....
Specificity              Alias for TrueNegativeRate
↪              |.....
Selectivity              Alias for TrueNegativeRate
↪              |.....
PositivePredictiveValue  TP / (TP + FP)
↪              |.....
Precision                Alias for PositivePredictiveValue
↪              |.....
NegativePredictiveValue  TN / (TN + FN)
↪              |.....
FalseNegativeRate        FN / (FN + TP)
↪              |.....
FalsePositiveRate        FP / (FP + TN)
↪              |.....
FalseDiscoveryRate       FP / (FP + TN)
↪              |.....
FalseOmissionRate        FP / (FP + TP)
↪              |.....
PositiveLikelihoodRatio  TPR / FPR
↪              |.....
NegativeLikelihoodRatio  FNR / TNR
↪              |.....
PrevalenceThreshold      sqrt(FPR) / (sqrt(FPR) + sqrt(TPR))
↪              |.....
ThreatScore              TP / (TP + FN + FP)
↪              |.....
F1Score                  2TP / (2TP + FN + FP)
↪              |.....
MeanAbsoluteError        sum(|y - x|) / n
↪              |.....
MeanSquaredError         sum((y - x)^2) / n
```

New metrics may appear in the list after importing a submodule. For example, `import neuronperf.torch` will register a new `topk` metric.

Custom Metrics

Simple Variants

If you wish to register a metric that is a slight tweak of an existing metric with different `init` args, you can use `register_metric_from_existing()`:

```
npf.register_metric_from_existing("topk", "topk_3", k=3)
```

This example registers a new metric `topk_3` from existing metric `topk`, passing `k=3` as at `init` time.

New Metrics

You can register your own metrics using `register_metric()`.

You metrics must extend `BaseEvalMetric`:

```
class BaseEvalMetric(ABC):
    """
    Abstract base class BaseEvalMetric from which other metrics inherit.
    """

    @abstractmethod
    def process_record(self, output: Any = None, target: Any = None) -> None:
        """Process an individual record and return the result."""
        pass

    @staticmethod
    def aggregate(metrics: Iterable["BaseEvalMetric"]) -> Any:
        """Combine a sequence of metrics into a single result."""
        raise NotImplementedError
```

For example:

```
import neuronperf as npf

class MyCustomMetric(npf.BaseEvalMetric):
    def __init__(self):
        super().__init__()
        self.passing = 0
        self.processed = 0

    def process_record(self, outputs, target):
        self.processed += 1
        if outputs == target:
            self.passing += 1

    @staticmethod
    def aggregate(metrics):
        passing = 0
        processed = 0
        for metric in metrics:
            passing += metric.passing
```

(continues on next page)

(continued from previous page)

```
            processed += metric.processed
        return passing / processed if processed else 0


npf.register_metric("MyCustomMetric", MyCustomMetric)
```

This document is relevant for: Inf1

This document is relevant for: Inf1

NeuronPerf Compile Guide

If you wish to compile multiple configurations at once, NeuronPerf provides a simplified and uniform API across frameworks. The output is a neuronperf_model_index that tracks the artifacts produces, and can be passed directly to the *benchmark* routine for a streamlined end-to-end process. This may be useful if you wish to test multiple configurations of your model on Neuron hardware.

You can manually specify the model index filename by passing `filename`, or let NeuronPerf generate one and return it for you. Compiled artifacts will be placed in a local `models` directory.

Please note that the default behavior will be to compile your model for each pipeline size, batch size, and cast mode in combination, which may take some time.

```
# Select a few batch sizes and pipeline configurations to test
batch_sizes = [1, 5, 10]
pipeline_sizes = [1, 2, 4]

# Construct example inputs
example_inputs = [torch.zeros([batch_size, 3, 224, 224], dtype=torch.float16) for batch_
↪size in batch_sizes]

# Compile all configurations
index = neuronperf.torch.compile(
   model,
   example_inputs,
   batch_sizes=batch_sizes,
   pipeline_sizes=pipeline_sizes,
)
```

If you wished to benchmark specific subsets of configurations, you could compile the specific configurations independently and later combine the results into a single index, as shown below.

```
# Compile with pipeline size 1 and vary batch dimension
batch_index = neuronperf.torch.compile(
   model,
   example_inputs,
   batch_sizes=batch_sizes,
   pipeline_sizes=1,
)

# Compile with batch size 1 and vary pipeline dimension
pipeline_index = neuronperf.torch.compile(
   model,
```

(continues on next page)

(continued from previous page)

```
    example_inputs[0],
    batch_sizes=1,
    pipeline_sizes=pipeline_sizes,
)

index = neuronperf.model_index.append(batch_index, pipeline_index)
neuronperf.model_index.save(index, 'model_index.json')
```

The `compile` function supports `batch_sizes`, `pipeline_sizes`, `cast_modes`, and custom `compiler_args`. If there is an error during compilation for a requested configuration, it will be logged and compilation will continue onward without terminating. (This is to support long-running compile jobs with many configurations.)

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Model Index Guide

A **model index** is a JSON file that tracks information about one or more compiled models. You can generate them using `compile`, by using the API described here, or you may create them manually in a text editor.

After a call to `compile` you may notice that you now have a `models` directory. You will also spot a new file named something like `model_83b3raj2.json` in your local directory, if you didn't provide a `filename` yourself.

A model index is not intended to be opaque; you should feel free to open, inspect, and modify it yourself. It contains some information about the artifacts that were compiled. Individual models referenced by the index can be handed to `benchmark` directly along with an example input, or you may pass the entire index as in the basic example above. Here is an example index:

```
python3 -m json.tool model_index.json
```

```
{
    "version": "0.0.0.0+0bc220a",
    "model_configs": [
        {
            "filename": "models/model_b1_p1_38793jda.pt",
            "input_idx": 0,
            "batch_size": 1,
            "pipeline_size": 1,
            "compile_s": 5.32
        }
    ]
}
```

An index is useful for keeping track of your compiled artifacts and their parameters. The advantages of using `neuronperf.[torch/tensorflow/mxnet].compile` are clearer when we wish to compile multiple variants of our model and benchmark all of them at the same time. All of the model artifacts and the index can be destroyed using `model_index.delete('model_index.json')`.

Benchmarking

When benchmarking with an index, there are some important details to keep in mind. If you originally built the index using a set of inputs, the model index has associated the `inputs` with the compiled models by their positional index.

For example:

```
batch_sizes = [1, 2]
inputs = [torch.zeros((b, 100)) for b in batch_sizes]
```

Here, `inputs[0]` corresponds to batch size 1. Therefore, the model index will contain a reference to input 0 for that model. When you call `benchmark`, you must pass inputs with the same shape in the same positions as at compile time.

Note: It's only necessary that there is an input with the correct shape at``inputs[input_index]``. The example data itself is not important.

Working with Indexes

The API detail below describes utilities for working with indexes. An `index` can be either a loaded index (JSON) or the path to an index (it will be loaded automatically).

Creating

```
index = neuronperf.model_index.create('/path/to/model', batch_size=1)
filename = neuronperf.model_index.save(index)
```

Once you have an index, you can pass its path directly to `benchmark`. You can also pass a custom filename instead:

```
index = neuronperf.model_index.create('/path/to/model', batch_size=1)
neuronperf.model_index.save(index, 'my_index.json')
```

Appending

If **multiple models use the same inputs**, you can append them together. For example, if you have the same batch size with multiple pipeline sizes, the inputs are the same, but the model changes.

```
pipeline_sizes = [1, 2, 3, 4]
indexes = [neuronperf.model_index.create(f'/path/to/model_p{p}', pipeline_size=p, batch_
↪size=5) for p in pipeline_sizes]
index = neuronperf.model_index.append(*indexes)
neuronperf.model_index.save(index, 'my_index.json')
```

Filtering

You can construct a new model index that is filtered by some parameter. For example, to get a new index with only batch sizes [1, 2], you could do:

```
new_index = neuronperf.model_index.filter(index, batch_sizes=[1, 2])
```

You can also benchmark subset of a model index by passing only the subset parameters of interest, but remember to ensure you provide the correct number of inputs for the index (even if some are not used).

For example, if you an index with models at `batch_sizes = [1, 2, 3]`, but only wish to benchmark batch size 2:

```
batch_sizes = [1, 2, 3]
inputs = [torch.zeros((b, 100)) for b in batch_sizes]
reports = neuronperf.torch.benchmark('model_index.json', inputs, batch_sizes=2)
```

Copying

You can copy an index to a new location with `neuronperf.model_index.copy(index, new_index_name, new_index_dir)`. This is mostly useful in combination with `filter`/`append`.

Deleting

If you wish to keep your compiled models, just delete the model index file yourself. If you want to delete your model index and all associated artifacts, use:

```
neuronperf.model_index.delete('my_index.json')
```

This document is relevant for: `Inf1`

13.4.3 NeuronPerf API Reference

This document is relevant for: `Inf1`

NeuronPerf API

Due to a bug in Sphinx, some of the type annotations may be incomplete. You can `download the source code here`. In the future, the source will be hosted in a more browsable way.

compile(*compile_fn*, *model*, *inputs*, *batch_sizes: Union[int, List[int]] = None*, *pipeline_sizes: Union[int, List[int]] = None*, *performance_levels: Union[str, List[int]] = None*, *models_dir: str = 'models'*, *filename: str = None*, *compiler_args: dict = None*, *verbosity: int = 1*, **args*, ***kwargs*) → str:

Compiles the provided model with each provided example input, pipeline size, and performance level. Any additional compiler_args passed will be forwarded to the compiler on every invocation.

Parameters

- **model** – The model to compile.
- **inputs** (*list*) – A list of example inputs.
- **batch_sizes** – A list of batch sizes that correspond to the example inputs.
- **pipeline_sizes** – A list of pipeline sizes to use. See *NeuronCore Pipeline*.

- **performance_levels** – A list of performance levels to try. Options are: 0 (max accuracy), 1, 2, 3 (max performance, default). See *Mixed precision and performance-accuracy tuning (neuron-cc)*.
- **models_dir** (*str*) – The directory where compilation artifacts will be stored.
- **model_name** (*str*) – An optional model name tag to apply to compiled artifacts.
- **filename** (*str*) – The name of the model index to write out. If not provided, a name will be generated and returned.
- **compiler_args** (*dict*) – Additional compiler arguments to be forwarded with every compilation.
- **verbosity** (*int*) – 0 = error, 1 = info, 2 = debug

Returns A model index filename. If a configuration fails to compile, it will not be included in the index and an error will be logged.

Return type str

benchmark(*load_fn: Callable[[str, int], Any], model_filename: str, inputs: Any, batch_sizes: Union[int, List[int]] = None, duration: float = BENCHMARK_SECS, n_models: Union[int, List[int]] = None, pipeline_sizes: Union[int, List[int]] = None, cast_modes: Union[str, List[str]] = None, workers_per_model: Union[int, None] = None, env_setup_fn: Callable[[int, Dict], None] = None, setup_fn: Callable[[int, Dict, Any], None] = None, preprocess_fn: Callable[[Any], Any] = None, postprocess_fn: Callable[[Any], Any] = None, dataset_loader_fn: Callable[[Any, int], Any] = None, verbosity: int = 1, multiprocess: bool = True, multiinterpreter: bool = False, return_timers: bool = False, device_type: str = 'neuron'*) → List[Dict]:

Benchmarks the model index or individiual model using the provided inputs. If a model index is provided, additional fields such as `pipeline_sizes` and `performance_levels` can be used to filter the models to benchmark. The default behavior is to benchmark all configurations in the model index.

Parameters

- **load_fn** – A function that accepts a model filename and device id, and returns a loaded model. This is automatically passed through the subpackage calls (e.g. `neuronperf.torch.benchmark`).
- **model_filename** (*str*) – A path to a model index from compile or path to an individual model. For CPU benchmarking, a class should be passed that can be instantiated with a default constructor (e.g. `MyModelClass`).
- **inputs** (*list*) – A list of example inputs. If the list contains tuples, they will be destructured on inference to support multiple arguments.
- **batch_sizes** – A list of ints indicating batch sizes that correspond to the inputs. Assumes 1 if not provided.
- **duration** (*float*) – The number of seconds to benchmark each model.
- **n_models** – The number of models to run in parallel. Default behavior runs 1 model and the max number of models possible, determined by a best effort from `device_type`, instance size, or other environment state.
- **pipeline_sizes** – A list of pipeline sizes to use. See *NeuronCore Pipeline*.
- **performance_levels** – A list of performance levels to try. Options are: 0 (max accuracy), 1, 2, 3 (max performance, default). See *Mixed precision and performance-accuracy tuning (neuron-cc)*.
- **workers_per_model** – The number of workers to use per model loaded. If `None`, this is automatically selected.
- **env_setup_fn** – A custom environment setup function to run in each subprocess before model loading. It will receive the benchmarker id and config.
- **setup_fn** – A function that receives the benchmarker id, config, and model to perform last minute configuration before inference.
- **preprocess_fn** – A custom preprocessing function to perform on each input before inference.
- **postprocess_fn** – A custom postprocessing function to perform on each input after inference.

- **multiprocess** (*bool*) – When True, model loading is dispatched to forked subprocesses. Should be left alone unless debugging.
- **multiinterpreter** (*bool*) – When True, benchmarking is performed in a new python interpreter per model. All parameters must be serializable. Overrides multiprocess.
- **return_timers** (*bool*) – When True, the return of this function is a list of tuples `(config, results)` with detailed information. This can be converted to reports with `get_reports(results)`.
- **stats_interval** (*float*) – Collection interval (in seconds) for metrics during benchmarking, such as CPU and memory usage.
- **device_type** (*str*) – This will be set automatically to one of the `SUPPORTED_DEVICE_TYPES`.
- **cost_per_hour** (*float*) – The price of this device / hour. Used to estimate cost / 1 million infs in reports.
- **model_name** (*str*) – A friendly name for the model to use in reports.
- **model_class_name** (*str*) – Internal use.
- **model_class_file** (*str*) – Internal use.
- **verbosity** (*int*) – 0 = error, 1 = info, 2 = debug

Returns A list of benchmarking results.

Return type list[dict]

get_reports(*results*)

Summarizes and combines the detailed results from `neuronperf.benchmark`, when run with `return_timers=True`. One report dictionary is produced per model configuration benchmarked. The list of reports can be fed directly to other reporting utilities, such as `neuronperf.write_csv`.

Parameters

- **results** (*list[tuple]*) – The list of results from `neuronperf.benchmark`.
- **batch_sizes** (*list[int]*) – The batch sizes that correspond to the *inputs* provided to `compile` and `benchmark`. Used to correct throughput values in the reports.

Returns A list of dictionaries that summarize the results for each model configuration.

Return type list[dict]

print_reports(*reports*, *cols=SUMMARY_COLS*, *sort_by='throughput_peak'*, *reverse=False*)

Print a report to the terminal. Example of default behavior:

```
>>> neuronperf.print_reports(reports)
throughput_avg latency_ms_p50 latency_ms_p99 n_models pipeline_size  workers_per_
↪model batch_size model_filename
329.667        6.073          6.109          1        1              2             ␣
↪   1          models/model_b1_p1_83bh3hhs.pt
```

Parameters

- **reports** – Results from *get_reports*.
- **cols** – The columns in the report to be displayed.
- **sort_by** – Sort the cols by the specified key.
- **reverse** – Sort order.

write_csv(*reports: list[dict]*, *filename: str = None*, *cols=REPORT_COLS*)

Write benchmarking reports to CSV file.

Parameters

- **reports** (*list[dict]*) – Results from *neuronperf.get_reports*.
- **filename** (*str*) – Filename to write. If not provided, generated from model_name in report and current timestamp.
- **cols** (*list[str]*) – The columns in the report to be kept.

Returns The filename written.

Return type str

write_json(*reports: list[dict]*, *filename: str = None*)

Writes benchmarking reports to a JSON file.

param list[dict] reports Results from *neuronperf.get_reports*.

param str filename Filename to write. If not provided, generated from model_name in report and current timestamp.

return The filename written.

rtype str

model_index.**append**(**model_indexes: Union[str, dict]*) → dict:

Appends the model indexes non-destructively into a new model index, without modifying any of the internal data.

This is useful if you have benchmarked multiple related models and wish to combine their respective model indexes into a single index.

Model name will be taken from the first index provided. Duplicate configs will be filtered.

Parameters **model_indexes** – Model indexes or paths to model indexes to combine.

Returns A new dictionary representing the combined model index.

Return type dict

model_index.**copy**(*old_index: Union[str, dict]*, *new_index: str*, *new_dir: str*) → str:

Copy an index to a new location. Will rename `old_index` to `new_index` and copy all model files into `new_dir`, updating the index paths.

This is useful for pulling individual models out of a pool.

Returns the path to the new index.

model_index.**create**(*filename*, *input_idx=0*, *batch_size=1*, *pipeline_size=1*, *cast_mode=DEFAULT_CAST*, *compile_s=None*)

Create a new model index from a pre-compiled model.

Parameters

- **filename** (*str*) – The path to the compiled model.
- **input_idx** (*int*) – The index in your inputs that this model should be run on.
- **batch_size** (*int*) – The batch size at compilation for this model.
- **pipeline_size** (*int*) – The pipeline size used at compilation for this model.
- **cast_mode** (*str*) – The casting option this model was compiled with.
- **compile_s** (*float*) – Seconds spent compiling.

Returns A new dictionary representing a model index.

Return type dict

model_index.delete(filename: str):

Deletes the model index and all associated models referenced by the index.

model_index.**filter**(*index: Union[str, dict]*, ***kwargs*) → dict:

Filters provided model index on provided criteria and returns a new index. Each kwarg is a standard (k, v) pair, where k is treated as a filter name and v may be one or more values used to filter model configs.

model_index.**load**(*filename*) → dict:

Load a NeuronPerf model index from a file.

model_index.**move**(*old_index: str*, *new_index: str*, *new_dir: str*) → str:

This is the same as `copy` followed by `delete` on the old index.

model_index.**save**(*model_index*, *filename: str = None*, *root_dir=None*) → str:

Save a NeuronPerf model index to a file.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

NeuronPerf Framework Notes

PyTorch

- **Requires: `torch-neuron`**
  - Versions: 1.7.x, 1.8.x, 1.9.x, 1.10.x
- Input to `compile`: `torch.nn.Module`
- Model inputs: `Any`.

TensorFlow 1.x

- **Requires: `tensorflow-neuron`**
  - Versions: All
- Input to `compile`: Path to uncompiled model dir from `saved_model.simple_save`
- Model inputs: Tensors must be provided as `numpy.ndarray`

Note: Although TensorFlow *tensors* must be `ndarray`, this doesn't stop you from wrapping them inside of data structures that traverse process boundaries safely. For example, you can still pass an input `dict` like `{'input_0': np.zeros((2, 1))}`.

TensorFlow 2.x

- **Requires: `tensorflow-neuron`**
  - Versions: All
- Input to `compile`: `tf.keras.Model`
- Model inputs: Tensors must be provided as `numpy.ndarray`

Note: Although TensorFlow *tensors* must be `ndarray`, this doesn't stop you from wrapping them inside of data structures that traverse process boundaries safely. For example, you can still pass an input `dict` like `{'input_0': np.zeros((2, 1))}`.

Apache MXNet (Incubating)

- **Requires: `mxnet-neuron`**
  - Versions 1.5, 1.8
- Input to `compile`: `tuple(sym, args, aux)`
- Inputs: Tensors must be provided as `mxnet.ndarray` or `numpy.ndarray`

This document is relevant for: `Inf1`

13.4.4 FAQ

This document is relevant for: `Inf1`

NeuronPerf FAQ

Table of contents

- *When should I use NeuronPerf?*
- *When should I **not** use NeuronPerf?*
- *Is NeuronPerf Open Source?*
- *What is the secret to obtaining the best numbers?*
- *What are the "best practices" that NeuronPerf uses?*

When should I use NeuronPerf?

When you want to measure the highest achievable performance for your model with Neuron.

When should I not use NeuronPerf?

When measuring end-to-end performance that includes your network serving stack. Instead, your should compare your e2e numbers to those obtained by NeuronPerf to optimize your serving overhead.

Is NeuronPerf Open Source?

Yes. You can `download the source here`.

What is the secret to obtaining the best numbers?

There is no secret sauce. NeuronPerf follows best practices.

What are the "best practices" that NeuronPerf uses?

- These vary slightly by framework and how your model was compiled
- For a model compiled for a single NeuronCore (DataParallel):
  - To maximize throughput, for `N` models, use `2 * N` worker threads
  - To minimize latency, use 1 worker thread per model
- Use a new Python process for each model to avoid GIL contention
- Ensure you benchmark long enough for your numbers to stabilize
- Ignore outliers at the start and end of inference benchmarking

This document is relevant for: `Inf1`

13.4.5 Troubleshooting

This document is relevant for: `Inf1`

NeuronPerf Troubleshooting

Table of contents

- *Compilation issues*
  - *Model fails to compile*
- *Benchmarking Issues*
  - *Benchmarking terminates early with errors*
- *Other Issues or Feature Requests*

Compilation issues

Model fails to compile

Please file a bug with as much information as possible.

Benchmarking Issues

Benchmarking terminates early with errors

- **Scroll up and read the output. Most likely causes are:**
  - invalid input shapes or
  - not enough memory to load the requested number of model copies on the device. Try passing `n_models=1` to `benchmark` again to test for memory issues.

Other Issues or Feature Requests

Please file a bug on Github.

This document is relevant for: `Inf1`

13.4.6 Release Notes

This document is relevant for: `Inf1`

What's New

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

Performance and Benchmarks Tools

- *NeuronPerf (Beta)*

Tutorials

TensorBoard

- *Neuron Plugin for TensorBoard (Trn1)*
- *Track Training Progress in TensorBoard using PyTorch Neuron*
- torch-neuronx-profiling-with-tb

System Tools

- track-system-monitor

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

FOURTEEN

SETUP GUIDE

This section walks you through the various options to install Neuron. You have to install Neuron on Trainium and Inferentia powered instances to enable deep-learning acceleration.

Inf1

PyTorch Neuron

TensorFlow Neuron

MXNet Neuron

Trn1

PyTorch Neuron

Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

FIFTEEN

DEPLOY CONTAINERS WITH NEURON

In this section you will find resources to help you use containers for your accelerated deep learning model acceleration on top of Inferentia and Trainium enabled instances.

The section is organized based on the target deployment environment and use case. In most cases, it is recommended to use a preconfigured Deep Learning Container (DLC) from AWS. Each DLC is pre-configured to have all of the Neuron components installed and is specific to the chosen ML Framework.

Locate Neuron DLC image

Latest Neuron DLC images

Find latest Neuron DLC images.

Locate specific Neuron DLC release

In the DLC release page do a search for Neuron to get the ECR repo location of specific Neuron DLC release.

Getting Started

Training

Launch Trn1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Trn1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Trn1 instances sizes and pricing see Trn1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

Install Drivers

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum install aws-neuronx-dkms-2.*  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import aws-efa-
↪installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig && gpg --
↪verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer
```

Install Docker

```
sudo yum install -y docker.io
sudo usermod -aG docker $USER

Logout and log back in to refresh membership.
```

Verify Docker

```
docker run hello-world
```

Expected result:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Verify Neuron Component

Once the environment is setup, a container can be started with –device=/dev/neuron# to specify desired set of Inferentia/Trainium devices to be exposed to the container. To find out the available neuron devices on your instance, use the command `ls /dev/neuron*`.

When running neuron-ls inside a container, you will only see the set of exposed Trainiums. For example:

```
docker run --device=/dev/neuron0 neuron-test neuron-ls

Would produce the following output in trn1.32xlarge:

::

   +--------+--------+--------+---------+
   | NEURON | NEURON | NEURON |   PCI   |
   | DEVICE | CORES  | MEMORY |   BDF   |
   +--------+--------+--------+---------+
   | 0      | 2      | 32 GB  | 10:1c.0 |
   +--------+--------+--------+---------+
```

Build and Run Docker Image

- how-to-build-neuron-container

Run Tutorial

Run training in Pytorch Neuron container

Inference

Launch Inf1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Install Drivers

```
# Configure Linux for Neuron repository updates
sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-NEURON.
↪PUB

# Update OS packages
sudo yum update -y

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to␣
↪latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum install aws-neuron-dkms -y

##########################################################################################
```

(continues on next page)

(continued from previous page)

```
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################
```

Install Docker

```
sudo yum install -y docker.io
sudo usermod -aG docker $USER

Logout and log back in to refresh membership.
```

Verify Docker

```
docker run hello-world
```

Expected result:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Verify Neuron Component

Once the environment is setup, a container can be started with –device=/dev/neuron# to specify desired set of Inferentia/Trainium devices to be exposed to the container. To find out the available neuron devices on your instance, use the command `ls /dev/neuron*`.

When running neuron-ls inside a container, you will only see the set of exposed Inferentias. For example:

```
docker run --device=/dev/neuron0 neuron-test neuron-ls

Would produce the following output in inf1.xlarge:
```

(continues on next page)

(continued from previous page)

```
::

   +--------------+---------+--------+-----------+-----------+------+------+
   |   PCI BDF    | LOGICAL | NEURON |  MEMORY   |  MEMORY   | EAST | WEST |
   |              |   ID    | CORES  | CHANNEL 0 | CHANNEL 1 |      |      |
   +--------------+---------+--------+-----------+-----------+------+------+
   | 0000:00:1f.0 |       0 |      4 | 4096 MB   | 4096 MB   |    0 |    0 |
   +--------------+---------+--------+-----------+-----------+------+------+
```

Run Tutorial

Run inference in pytorch neuron container

Kubernetes Getting Started

Prerequisite

- Working kubernetes cluster
- Inf1/Trn1 instances as worker nodes with attached roles allowing: * ECR read access policy to retrieve container images from ECR: **arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly**
- tutorial-docker-env-setup: to install required packages in the worker nodes. With EKS, the EKS optimized accelarated AMI has the necessary neuron components installed
- Kubernetes node object has instance-type set to inf1/trn1 types. For ex, `"node.kubernetes.io/instance-type": "inf1.2xlarge"` or `"node.kubernetes.io/instance-type": "trn1.2xlarge"`

Deploy Neuron Device Plugin

Neuron device plugin exposes Neuron cores & devices to kubernetes as a resource. aws.amazon.com/neuroncore, aws.amazon.com/neurondevice, aws.amazon.com/neuron are the resources that the neuron device plugin registers with the kubernetes. aws.amazon.com/neuroncore is used for allocating neuron cores to the container. aws.amazon.com/neurondevice is used for allocating neuron devices to the container. When neurondevice is used all the cores belonging to the device will be allocated to container. aws.amazon.com/neuron also allocates neurondevices and this exists just to be backward compatible with already existing installations. aws.amazon.com/neurondevice is the recommended resource for allocating devices to the container.

- Make sure prequisite are satisified
- Download the neuron device plugin yaml file. `k8s-neuron-device-plugin.yml`
- Download the neuron device plugin rbac yaml file. This enables permissions for device plugin to update the node and Pod annotations. `k8s-neuron-device-plugin-rbac.yml`
- Apply the Neuron device plugin as a daemonset on the cluster with the following command

  ```
  kubectl apply -f k8s-neuron-device-plugin-rbac.yml
  kubectl apply -f k8s-neuron-device-plugin.yml
  ```

- Verify that neuron device plugin is running

```
kubectl get ds neuron-device-plugin-daemonset --namespace kube-system
```

Expected result (with 2 nodes in cluster):

```
NAME                              DESIRED   CURRENT   READY   UP-TO-DATE  ␣
↪AVAILABLE   NODE SELECTOR   AGE
neuron-device-plugin-daemonset    2         2         2       2           ␣
↪2           <none>          27h
```

- Verify that the node has allocatable neuron cores and devices with the following command

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,NeuronCore:.
↪status.allocatable.aws\.amazon\.com/neuroncore"
```

Expected result:

```
NAME                                          NeuronCore
ip-192-168-65-41.us-west-2.compute.internal   32
ip-192-168-87-81.us-west-2.compute.internal   32
```

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,NeuronDevice:.
↪status.allocatable.aws\.amazon\.com/neurondevice"
```

Expected result:

```
NAME                                          NeuronDevice
ip-192-168-65-41.us-west-2.compute.internal   16
ip-192-168-87-81.us-west-2.compute.internal   16
```

Deploy Neuron Scheduler Extension

Neuron scheduler extension is required for scheduling pods that require more than one Neuron core or device resource. Refer k8s-neuron-scheduler-flow for details on how the neuron scheduler extension works. Neuron scheduler extension filter out nodes with non-contiguous core/device ids and enforces allocation of contiguous core/device ids for the PODs requiring it.

Multiple Scheduler Approach

In cluster environments where there is no access to default scheduler, the neuron scheduler extension can be used with another scheduler. A new scheduler is added (along with the default scheduler) and then the pod's that needs to run the neuron workload use this new scheduler. Neuron scheduler extension is added to this new scheduler. EKS natively does not yet support the neuron scheduler extension and so in the EKS environment this is the only way to add the neuron scheduler extension.

- Make sure Neuron device plugin is running
- Download the my scheduler `my-scheduler.yml`
- Download the scheduler extension `k8s-neuron-scheduler-eks.yml`
- Apply the neuron-scheduler-extension

```
kubectl apply -f k8s-neuron-scheduler-eks.yml
```

- Get the neuron scheduler extension cluster ip

```
kubectl get service --namespace kube-system k8s-neuron-scheduler -o␣
↪jsonpath='{.spec.clusterIP}'
```

- Modify the urlPrefix in my-scheduler.yml. Replace 127.0.0.1 with cluster ip above.

```
"urlPrefix": "http://127.0.0.1:12345 to "urlPrefix": "http://<cluster-ip-
↪from-above>:12345
```

- Apply the modified my-scheduler.yml

```
kubectl apply -f my-scheduler.yml
```

- Check there are no errors in the my-scheduler pod logs and the k8s-neuron-scheduler pod is bound to a node

```
kubectl logs -n kube-system my-scheduler-79bd4cb788-hq2sq
```

```
I1012 15:30:21.629611       1 scheduler.go:604] "Successfully bound pod to␣
↪node" pod="kube-system/k8s-neuron-scheduler-5d9d9d7988-xcpqm" node="ip-
↪192-168-2-25.ec2.internal" evaluatedNodes=1 feasibleNodes=1
```

- When running new pod's that need to use the neuron scheduler extension, make sure it uses the my-scheduler as the scheduler. Sample pod spec is below

```
apiVersion: v1
kind: Pod
metadata:
name: <POD_NAME>
spec:
restartPolicy: Never
schedulerName: my-scheduler
containers:
    - name: <POD_NAME>
    command: ["<COMMAND>"]
    image: <IMAGE_NAME>
    resources:
        limits:
        cpu: "4"
        memory: 4Gi
        aws.amazon.com/neuroncore: 9
        requests:
        cpu: "1"
        memory: 1Gi
```

- Once the neuron workload pod is run, make sure logs in the k8s neuron scheduler has successfull filter/bind request

```
kubectl logs -n kube-system k8s-neuron-scheduler-5d9d9d7988-xcpqm
```

```
2022/10/12 15:41:16 POD nrt-test-5038 fits in Node:ip-192-168-2-25.ec2.
↪internal
2022/10/12 15:41:16 Filtered nodes: [ip-192-168-2-25.ec2.internal]
```

(continues on next page)

(continued from previous page)

```
2022/10/12 15:41:16 Failed nodes: map[]
2022/10/12 15:41:16 Finished Processing Filter Request...
```

```
2022/10/12 15:41:16 Executing Bind Request!
2022/10/12 15:41:16 Determine if the pod %v is NeuronDevice podnrt-test-
↪5038
2022/10/12 15:41:16 Updating POD Annotation with alloc devices!
2022/10/12 15:41:16 Return aws.amazon.com/neuroncore
2022/10/12 15:41:16 neuronDevUsageMap for resource:aws.amazon.com/
↪neuroncore in node: ip-192-168-2-25.ec2.internal is [false false false␣
↪false false false false false false false false false false false␣
↪false]
2022/10/12 15:41:16 Allocated ids for POD nrt-test-5038 are: 0,1,2,3,4,5,6,
↪7,8
2022/10/12 15:41:16 Try to bind pod nrt-test-5038 in default namespace to␣
↪node ip-192-168-2-25.ec2.internal with &Binding{ObjectMeta:{nrt-test-
↪5038    8da590b1-30bc-4335-b7e7-fe574f4f5538  0 0001-01-01 00:00:00␣
↪+0000 UTC <nil> <nil> map[] map[] [] []  []},Target:ObjectReference{Kind:
↪Node,Namespace:,Name:ip-192-168-2-25.ec2.internal,UID:,APIVersion:,
↪ResourceVersion:,FieldPath:,},}
2022/10/12 15:41:16 Updating the DevUsageMap since the bind is successful!
2022/10/12 15:41:16 Return aws.amazon.com/neuroncore
2022/10/12 15:41:16 neuronDevUsageMap for resource:aws.amazon.com/
↪neuroncore in node: ip-192-168-2-25.ec2.internal is [false false false␣
↪false false false false false false false false false false false␣
↪false]
2022/10/12 15:41:16 neuronDevUsageMap for resource:aws.amazon.com/
↪neurondevice in node: ip-192-168-2-25.ec2.internal is [false false false␣
↪false]
2022/10/12 15:41:16 Allocated devices list 0,1,2,3,4,5,6,7,8 for resource␣
↪aws.amazon.com/neuroncore
2022/10/12 15:41:16 Allocated devices list [0] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [0] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [0] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [0] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [1] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [1] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [1] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [1] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Allocated devices list [2] for other resource aws.
↪amazon.com/neurondevice
2022/10/12 15:41:16 Return aws.amazon.com/neuroncore
2022/10/12 15:41:16 Succesfully updated the DevUsageMap [true true true␣
↪true true true true true true false false false false false false false]␣
↪ and otherDevUsageMap [true true true false] after alloc for node ip-192-
↪168-2-25.ec2.internal
```

(continued on next page)

(continued from previous page)

```
2022/10/12 15:41:16 Finished executing Bind Request...
```

Default Scheduler Approach

- Make sure Neuron device plugin is running
- Download the scheduler config map `k8s-neuron-scheduler-configmap.yml`
- Download the scheduler extension `k8s-neuron-scheduler.yml`
- Enable the kube-scheduler with option to use configMap for scheduler policy. In your cluster.yml Please update the spec section with the following

```
spec:
  kubeScheduler:
  usePolicyConfigMap: true
```

- Launch the cluster

```
kops create -f cluster.yml
kops create secret --name neuron-test-1.k8s.local sshpublickey admin -i ~/.
↪ssh/id_rsa.pub
kops update cluster --name neuron-test-1.k8s.local --yes
```

- Apply the k8s-neuron-scheduler-configmap.yml [Registers neuron-scheduler-extension with kube-scheduler]

```
kubectl apply -f k8s-neuron-scheduler-configmap.yml
```

- Launch the neuron-scheduler-extension

```
kubectl apply -f k8s-neuron-scheduler.yml
```

Tutorials

Training

This document is relevant for: `Trn1`

15.1 Run training in Pytorch Neuron container

Table of Contents

- *Overview*
- *Setup Environment*

15.1.1 Overview

This tutorial demonstrates how to run a pytorch container on an trainium instance.

By the end of this tutorial you will be able to run simple mlp training using the container

You will use an trn1.2xlarge to test your Docker configuration for Trainium.

To find out the available neuron devices on your instance, use the command `ls /dev/neuron*`.

15.1.2 Setup Environment

1. **Launch an Trn1 Instance**

   - Please follow the instructions at launch an Amazon EC2 Instance to Launch an Trn1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Trn1 instances sizes and pricing see Trn1 web page.
   - Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10.
   - When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.
   - After launching the instance, follow the instructions in Connect to your instance to connect to the instance

   Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

   To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

   Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

   If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

2. Set up docker environment according to tutorial-docker-env-setup

3. A sample Dockerfile for for torch-neuron can be found here trainium-dlc-dockerfile. This dockerfile needs the mlp train script found here mlp-train

With the files in a dir, build the image with the following command:

```
docker build . -f Dockerfile.pt -t neuron-container:pytorch
```

Run the following command to start the container

```
docker run -it --name pt-cont --net=host --device=/dev/neuron0 neuron-container:pytorch␣
↪python3 /opt/ml/mlp_train.py
```

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

15.2 Deploy a simple mlp training script as a Kubernetes job

This tutorial uses mlp train as a teaching example on how to deploy an training application using Kubernetes on the Trn1 instances.

15.2.1 Prerequisite:

- tutorial-k8s-env-setup-for-neuron: to setup k8s support on your cluster.
- Trn1 instances as worker nodes with attached roles allowing:
  - ECR read access policy to retrieve container images from ECR: **arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly**
- Have a container image that is build using *Run training in Pytorch Neuron container*

15.2.2 Deploy a mlp training image

1. Create a file named *mlp_train.yaml* with the contents below.

Note: In the image: add the appropriate location of the image

```
apiVersion: v1
kind: Pod
metadata:
  name: trn1-mlp
spec:
  restartPolicy: Never
  schedulerName: default-scheduler
  hostNetwork: true
  nodeSelector:
    beta.kubernetes.io/instance-type: trn1.32xlarge
    beta.kubernetes.io/instance-type: trn1.2xlarge
  containers:
    - name: trn1-mlp
      command: ["/usr/local/bin/python3"]
      args:  ["/opt/ml/mlp_train.py"]
      image: 647554078242.dkr.ecr.us-east-1.amazonaws.com/sunda-pt:k8s_mlp_0907
      imagePullPolicy: IfNotPresent
      env:
      - name: NEURON_RT_LOG_LEVEL
        value: "INFO"
      resources:
        limits:
          aws.amazon.com/neuron: 2
        requests:
          aws.amazon.com/neuron: 2
```

2. Deploy the pod.

```
kubectl apply -f mlp_train.yaml
```

3. Check the logs to make sure training completed

```
kubectl logs <pod name>

Your log should have the following
```

```
Final loss is 0.1977
----------End Training ---------------
```

This document is relevant for: `Trn1`

Inference

This document is relevant for: `Inf1`

15.3 Run inference in pytorch neuron container

Table of Contents

- *Overview*
- *Setup Environment*

15.3.1 Overview

This tutorial demonstrates how to run a pytorch DLC on an inferentia instance.

By the end of this tutorial you will be able to run the inference using the container

You will use an inf1.2xlarge to test your Docker configuration for Inferentia.

To find out the available neuron devices on your instance, use the command `ls /dev/neuron*`.

15.3.2 Setup Environment

1. **Launch an Inf1 Instance**
    - Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
    - Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Amazon Linux 2 AMI(HVM) - Kernel 5.10
    - After launching the instance, follow the instructions in Connect to your instance to connect to the instance
2. Set up docker environment according to tutorial-docker-env-setup

3. A sample Dockerfile for for torch-neuron can be found here inference-dlc-dockerfile. This dockerfile needs the torchserve entrypoint found here torchserve-neuron and torchserve config.properties found here torchserve-config-properties.

With the files in a dir, build the image with the following command:

```
docker build . -f Dockerfile.pt -t neuron-container:pytorch
```

Run the following command to start the container

```
docker run -itd --name pt-cont -p 80:8080 -p 8081:8081 --device=/dev/neuron0 neuron-
↪container:pytorch /usr/local/bin/entrypoint.sh -m 'pytorch-resnet-neuron=https://aws-
↪dlc-sample-models.s3.amazonaws.com/pytorch/Resnet50-neuron.mar' -t /home/model-server/
↪config.properties
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

15.4 Deploy a TensorFlow Resnet50 model as a Kubernetes service

This tutorial uses Resnet50 model as a teaching example on how to deploy an inference application using Kubernetes on the Inf1 instances.

15.4.1 Prerequisite:

- tutorial-k8s-env-setup-for-neuron: to setup k8s support on your cluster.
- Inf1 instances as worker nodes with attached roles allowing:
  - ECR read access policy to retrieve container images from ECR: **arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly**
  - S3 access to retrieve saved_model from within tensorflow serving container.

15.4.2 Deploy a TensorFlow Serving application image

A trained model must be compiled to an Inferentia target before it can be deployed on Inferentia instances. To continue, you will need a Neuron optimized TensorFlow model saved in Amazon S3. If you don't already have a SavedModel, please follow the tutorial for creating a Neuron compatible ResNet50 model and upload the resulting SavedModel to S3.

ResNet-50 is a popular machine learning model used for image classification tasks. For more information about compiling Neuron models, see The AWS Inferentia Chip With DLAMI in the AWS Deep Learning AMI Developer Guide.

The sample deployment manifest manages a pre-built inference serving container for TensorFlow provided by AWS Deep Learning Containers. Inside the container is the AWS Neuron Runtime and the TensorFlow Serving application. A complete list of pre-built Deep Learning Containers optimized for Neuron is maintained on GitHub under Available Images. At start-up, the DLC will fetch your model from Amazon S3, launch Neuron TensorFlow Serving with the saved model, and wait for prediction requests.

The number of Neuron devices allocated to your serving application can be adjusted by changing the *aws.amazon.com/neuron* resource in the deployment yaml. Please note that communication between TensorFlow Serving and the Neuron runtime happens over GRPC, which requires passing the *IPC_LOCK* capability to the container.

1. Create a file named *rn50_deployment.yaml* with the contents below. Update the region-code and model path to match your desired settings. The model name is for identification purposes when a client makes a request to the TensorFlow server. This example uses a model name to match a sample ResNet50 client script that will be used in a later step for sending prediction requests.

Note:

1. Replace the s3 bucket name in model_base_path arg in the file with the location of the where the saved model was stored in s3.
2. In the image: add the appropriate location of the DLC tensorflow image

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: k8s-neuron-test
  labels:
    app: k8s-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: k8s-neuron-test
        role: master
    spec:
      containers:
        - name: k8s-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
↪neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://your-bucket-of-models/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
```

(continues on next page)

(continued from previous page)

```
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

2. Deploy the model.

```
kubectl apply -f rn50_deployment.yaml
```

3. Create a file named *rn50_service.yaml* with the following contents. The HTTP and gRPC ports are opened for accepting prediction requests.

```
kind: Service
apiVersion: v1
metadata:
  name: k8s-neuron-test
  labels:
    app: k8s-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: k8s-neuron-test
    role: master
```

4. Create a Kubernetes service for your TensorFlow model Serving application.

```
kubectl apply -f rn50_service.yaml
```

15.4.3 Make predictions against your TensorFlow Serving service

1. To test locally, forward the gRPC port to the *k8s-neuron-test* service.

```
kubectl port-forward service/k8s-neuron-test 8500:8500 &
```

2. Create a Python script called *tensorflow-model-server-infer.py* with the following content. This script runs inference via gRPC, which is service framework.

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/
↪kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))
```

3. Run the script to submit predictions to your service.

```
python3 tensorflow-model-server-infer.py

Your output should look like the following:
```

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613), (u'n02123159
↪', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat', 0.063844085), (u
↪'n02128757', u'snow_leopard', 0.009240591)]]
```

This document is relevant for: `Inf1`

Developer Flows

Inference

This document is relevant for: `Inf1`

15.5 Deploy Neuron Container on EC2

Table of Contents

- *Description*
- *Setup Environment*

15.5.1 Description

You can use the Neuron version of the AWS Deep Learning Containers to run inference on inf1 instances. In this developer flow, you provision an EC2 inf1 instance using a Deep Learning AMI (DLAMI), pull the container image with the Neuron version of the desired framework, and run the container as a server for the already compiled model. This developer flow assumes the model has already has been compiled through a compilation developer flow

15.5.2 Setup Environment

1. **Launch an Inf1 Instance**
   - Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
   - Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Ubuntu 18 AMI or Amazon Linux 2 AMI, you can also choose Ubuntu 18 or Amazon Linux 2 Deep Learning AMI (DLAMI)
   - After launching the instance, follow the instructions in Connect to your instance to connect to the instance
2. Once you have your EC2 environment set according to tutorial-docker-env-setup, you can build and run a Neuron container using the how-to-build-neuron-container section above.

Note: Prior to running the container, make sure that the Neuron runtime on the instance is turned off, by running the command:

```
sudo service neuron-rtd stop
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

15.6 Deploy Neuron Container on Elastic Container Service (ECS)

Table of Contents

- *Description*
- *Setup Environment*

15.6.1 Description

You can use the Neuron version of the AWS Deep Learning Containers to run inference on Amazon Elastic Container Service (ECS). In this developer flow, you set up an ECS cluster with inf1 instances, create a task description for your inference service and deploy it to your cluster. This developer flow assumes:

1. The model has already been compiled through *Compilation with Framework API on EC2 instance* or through *Compilation with Sagemaker Neo*.
2. You already set up your container to retrieve it from storage.

15.6.2 Setup Environment

1. **Set up an Amazon ECS cluster:** Follow the instructions on Setting up Amazon ECS for Deep Learning Containers
2. **Define an Inference Task:** Use the instruction on the DLC Inference on ECS Tutorial to define a task and create a service for the appropriate framework.

   When creating tasks for inf1 instances on ECS, be aware of the considerations and requirements listed in Working with inference workloads on Amazon ECS.
3. Use the container image created using how-to-build-neuron-container as the `image` in your task definition.

Note: Before deploying your task definition to your ECS cluster, make sure to push the image to ECR. Refer to Pushing a Docker image for more information.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

15.7 Deploy Neuron Container on Elastic Kubernetes Service (EKS)

Table of Contents

- *Description*
- *Setup Environment*
  - *Self-managed Kubernetes*

15.7.1 Description

You can use the Neuron version of the AWS Deep Learning Containers to run inference on Amazon Elastic Kubernetes Service (EKS). In this developer flow, you set up an EKS cluster with Inf1 instances, create a Kubernetes manifest for your inference service and deploy it to your cluster. This developer flow assumes:

1. The model has already been compiled through *Compilation with Framework API on EC2 instance* or through *Compilation with Sagemaker Neo*.
2. You already set up your container to retrieve it from storage.

15.7.2 Setup Environment

1. **Install pre-requisits:** Follow these instruction to install or upgrade the *eksctl* command line utility on your local computer.

   Follow these instruction to install *kubectl* in the same computer. *kubectl* is a command line tool for working with Kubernetes clusters.

2. Follow the instructions in this EKS documentation link to set up AWS Inferentia on your EKS cluster. Using the YML deployment manifest shown in the same link, replace the *image* in the *containers* specification with the one you built using how-to-build-neuron-container above.

   Note: Before deploying your task definition to your EKS cluster, make sure to push the image to ECR. Refer to Pushing a Docker image for more information.

Self-managed Kubernetes

Please refer to tutorial-k8s-env-setup-for-neuron. In *Deploy a TensorFlow Resnet50 model as a Kubernetes service*, the container image referenced in the YML manifest is created using how-to-build-neuron-container.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

15.8 Bring Your Own Neuron Container to Sagemaker Hosting

Table of Contents

- *Description*
- *Setup Environment*

15.8.1 Description

You can use a SageMaker Notebook or an EC2 instance to compile models and build your own containers for deployment on SageMaker Hosting using ml.inf1 instances. In this developer flow, you provision a Sagemaker Notebook or an EC2 instance to train and compile your model to Inferentia. Then you deploy your model to SageMaker Hosting using the SageMaker Python SDK. Follow the steps bellow to setup your environment. Once your environment is set you'll be able to follow the *BYOC HuggingFace pretrained BERT container to Sagemaker Tutorial* .

15.8.2 Setup Environment

1. **Create a Compilation Instance:** If using an **EC2 instance for compilation** you can use an Inf1 instance to compile and test a model. Follow these steps to launch an Inf1 instance:

   - Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
   - Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Ubuntu 18 AMI or Amazon Linux 2 AMI, you can also choose Ubuntu 18 or Amazon Linux 2 Deep Learning AMI (DLAMI)
   - After launching the instance, follow the instructions in Connect to your instance to connect to the instance

   If using an **SageMaker Notebook for compilation**, follow the instructions in Get Started with Notebook Instances to provision the environment.

   It is recommended that you start with an ml.c5.4xlarge instance for the compilation. Also, increase the volume size of you SageMaker notebook instance, to accomodate the models and containers built locally. A volume of 10GB is sufficient.

   > **Note:** To compile the model in the SageMaker Notebook instance, you'll need to update the conda environments to include the Neuron Compiler and Neuron Framework Extensions. Follow the installation guide on the section how-to-update-to-latest-Neuron-Conda-Env to update the environments.

2. **Set up the environment to compile a model, build your own container and deploy:** To compile your model on EC2 or SageMaker Notebook, follow the *Set up a development environment* section on the EC2 *Setup Environment* documentation.

   Refer to Adapting Your Own Inference Container documentation for information on how to bring your own containers to SageMaker Hosting.

   Make sure to add the **AmazonEC2ContainerRegistryPowerUser** role to your IAM role ARN, so you're able to build and push containers from your SageMaker Notebook instance.

   > **Note:** The container image can be created using how-to-build-neuron-container.

This document is relevant for: `Inf1`

FAQ, Troubleshooting and Release Note

This document is relevant for: `Inf1`, `Trn1`

15.9 Neuron Containers FAQ

Table of Contents

- *Where can I find DLC images*
- *What is OCI Neuron Hook and do we need that*
- *What container runtimes are supported*
- *How to expose Neuron Devices to Container*
- *How to expose Neuron Cores to Container*
- *Can Neuron Devices be shared by different Containers running in the same Host*
- *Can Neuron Cores be shared by different Containers running in the same Host*
- *When would you use Neuron K8 Scheduler Extension*
- *How to add EFA devices to the container*
- *Can distributed training jobs be run without EFA devices in container*

15.9.1 Where can I find DLC images

- The Inference/Training DLC images can be found here.
- In the DLC release page do a search for neuron to get the ECR repo location of specific neuron DLC release.

15.9.2 What is OCI Neuron Hook and do we need that

Neuron devices are exposed to the containers using the –device option in the docker run command. Docker runtime (runc) does not yet support the ALL option to expose all neuron devices to the container.

With OCI neuron hook support is added to expose ALL devices to container using an environment variable, "AWS_NEURON_VISIBLE_DEVICES=ALL". For more details please refer oci neuron hook

In Kubernetes, if we are using the device plugin version 1.7 & below, then the oci neuron hook is needed. If using device plugin version >= 1.8 then oci neuron hook is not needed

15.9.3 What container runtimes are supported

Neuron containers have been tested to work with docker, containerd, cri-o runtimes without any changes. If the oci neuron hook is used then they need to be enabled in the runtime config. For more details please refer oci neuron hook

15.9.4 How to expose Neuron Devices to Container

Neuron Device: Represents the number of Inferentia/Trainium chips in the instance. Refer Container Devices for more details

15.9.5 How to expose Neuron Cores to Container

Neuron Core: Represents the number of Neuron Cores in the instance. Refer Container Cores for more details. Each Inferentia Chip has 4 Neuron Cores and each Trainium chip has 2 Neuron Cores. When the devices are exposed to the containers all the cores in the device are available for use in the container. Please refer *Neuron Runtime Configuration* to see how the environment variables NEURON_RT_VISIBLE_CORES and NEURON_RT_NUM_CORES can be used to assign core to containers

15.9.6 Can Neuron Devices be shared by different Containers running in the same Host

Yes, except in Kubernetes environment where the devices cannot be shared

15.9.7 Can Neuron Cores be shared by different Containers running in the same Host

No

15.9.8 When would you use Neuron K8 Scheduler Extension

The neuron cores/devices that are exposed to the container needs to be contiguous. The kubernetes device plugin does not guarantee the devices to be contiguous. The K8 Neuron Scheduler Extension takes care of assigning contiguous devices to the containers.

15.9.9 How to add EFA devices to the container

The EFA devices are exposed to the container using the –device option

```
--device /dev/infiniband/uverbs0
```

In the kubernetes environment the EFA device plugin is used to detect and advertise EFA interfaces.

:: kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/manifest/efa-k8s-device-plugin.yml

Application can use the resource type vpc.amazonaws.com/efa in a pod request spec

```
vpc.amazonaws.com/efa: 4
```

15.9.10 Can distributed training jobs be run without EFA devices in container

No. For distributed training jobs in Trainium all the EFA inrerfaces in trn1.32xlarge needs to be exposed to the containers

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

15.10 Troubleshooting Neuron Containers

This document aims to provide more information on how to fix issues you might encounter while using the Neuron Containers. For each issue we will provide an explanation of what happened and what can potentially correct the issue.

If your issue is not listed below or you have a more nuanced problem, contact us via issues posted to this repo, the AWS Neuron developer forum, or through AWS support.

Neuron Container includes the following Neuron Components. For issues relating to these components inside the container refer the individual component troubleshooting guides *Troubleshooting Guide*

- Neuron Runtime/Driver
- Pytorch/Tenosrflow/MXNet frameworks
- Libfabric/EFA

The following are container specific issues

15.10.1 Neuron Device Not found

The neuron container expects the neuron devices to be exposed to the container as referenced in container-devices.

Please look at the container logs to see messages like below

```
2022-Sep-08 17:55:23.0768    19:19    ERROR  TDRV:tdrv_get_dev_info
↪ No neuron device available
```

If the above message is seen then devices are not exposed to container

Solution

- Refer container-devices and make sure the devices are exposed to container
- If specific cores are being used refer container-cores and make sure the cores are exposed to container
- In kubernetes environment refer k8s-specify-devices or k8s-specify-cores to make sure neuron devices/cores are there in pods container spec

15.10.2 Contiguous Device ID's

Neuron runtime expects the inferentia/trainium device id's to be contigious. If the device id's are not contiguous you might see error messages like below

```
2022-Sep-08 21:52:11.0307     7:7     ERROR  TDRV:tdrv_init_mla_phase1
↪ Could not open the nd1
```

```
2022-Sep-08 23:00:05.0667     8:8     ERROR   NRT:nrt_allocate_neuron_cores
↪ Neuron cores are not contiguous
```

Solution

- In the docker run command make sure the devices specified using –device are all contiguous
- If oci neuron hook is used and the env variable AWS_NEURON_VISIBLE_DEVICES is used then make sure the

devices specified are all contiguous * In kubernetes environment with just the neuron device plugin running there is no guarantee that the devices allocated will be contiguous. Make sure to run the neuron scheduler extension as specified in neuron-k8-scheduler-ext

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

15.11 Neuron Containers Release Notes

Table of contents

- *Neuron 2.5.0*
- *Neuron 2.4.0*
- *Neuron 2.3.0*
- *Neuron 1.19.0*
- *Neuron 1.16.0*

15.11.1 Neuron 2.5.0

Date: 11/07/2022

- Neuron now supports trn1-based training in Sagemaker and Deep Learning Containers using PyTorch. Find Neuron DLC containers here: https://github.com/aws/deep-learning-containers/blob/master/available_images.md#neuron-containers

15.11.2 Neuron 2.4.0

Date: 10/27/2022

- Neuron now supports Kubernetes work scheduling at the level of NeuronCore. Updates on how to use the new core allocation method is captured in the Kubernetes documentation on this site.

15.11.3 Neuron 2.3.0

Date: 10/10/2022

- Now supporting TRN1 and INF1 EC2 instance types as part of Neuron. There is an optional aws-neuronx-oci-hooks package users may install for conveince that supports use of the AWS_NEURON_VISIBLE_DEVICES environment variable when launching containers. New DLC containers will be coming soon in support of training workloads on TRN1.

15.11.4 Neuron 1.19.0

Date: 04/29/2022

- Neuron Kubernetes device driver plugin now can figure out communication with the Neuron driver without the *oci hooks*. Starting with *Neuron 1.19.0* release, installing `aws-neuron-runtime-base` and `oci-add-hooks` are no longer a requirement for Neuron Kubernetes device driver plugin.

15.11.5 Neuron 1.16.0

Date: 10/27/2021

New in this release

- Starting with Neuron 1.16.0, use of Neuron ML Frameworks now comes with an integrated Neuron Runtime as a library, as a result it is no longer needed to deploy `neuron-rtd`. Please visit neuron-containers for more information.
- When using containers built with components from Neuron 1.16.0, or newer, please use `aws-neuron-dkms` version 2.1 or newer and the latest version of `aws-neuron-runtime-base`. Passing additional system capabilities is no longer required.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

15.12 Neuron K8 Release Notes

Table of contents

- *Introduction*
- *Neuron K8 release [2.1.0.0]*
- *Neuron K8 release [2.0.0.0]*

- *Neuron K8 release [1.9.3.0]*
- *Neuron K8 release [1.9.2.0]*
- *Neuron K8 release [1.9.0.0]*
- *Neuron K8 release [1.8.2.0]*
- *Neuron K8 release [1.7.7.0]*
- *Neuron K8 release [1.7.3.0]*
- *[1.6.22.0]*
- *[1.6.15.0]*
- *[1.6.7.0]*
- *[1.6.0.0]*
- *[1.5.3.0]*
- *[1.4.1.0]*
- *[1.3.2.0]*
- *[1.2.0.0]*
- *[1.1.23.0]*
- *[1.1.17.0]*
- *[1.0.11000.0]*

15.12.1 Introduction

This document lists the current release notes for AWS Neuron Kubernetes (k8) components. Neuron K8 components include a device plugin and a scheduler extension to assist with deployment and management of inf1 nodes within Kubernetes clusters. Both components are offered as pre-built containers in Public ECR and ready for deployment.

- **Device Plugin:** public.ecr.aws/neuron/neuron-device-plugin:2.0.0.0
- **Neuron Scheduler:** public.ecr.aws/neuron/neuron-scheduler:2.0.0.0

It's recommended to pin the version of the components used and to never use the "latest" tag. To get the list of image tags, please refer to these notes or check the image tags on the repo directly.

To Pull the Images from ECR:

```
docker pull  public.ecr.aws/neuron/neuron-device-plugin:2.0.0.0
docker pull  public.ecr.aws/neuron/neuron-scheduler:2.0.0.0
```

15.12.2 Neuron K8 release [2.1.0.0]

Date: 10/27/2022

Summary

- Added support for NeuronCore based scheduling to the Neuron Kubernetes Scheduler. Learn more about how to use NeuronCores for finer grain control over container scheduling by following the K8 tutorials documentation in the *containers section*.

15.12.3 Neuron K8 release [2.0.0.0]

Date: 10/10/2022

Summary

- Added support for TRN1 and INF1 EC2 instance types.

15.12.4 Neuron K8 release [1.9.3.0]

Date: 08/02/2022

Summary

- Minor updates.

15.12.5 Neuron K8 release [1.9.2.0]

Date: 05/27/2022

Summary

- Minor updates.

15.12.6 Neuron K8 release [1.9.0.0]

Date: 04/29/2022

Summary

- Minor updates.

15.12.7 Neuron K8 release [1.8.2.0]

Date: 03/25/2022

Summary

- Minor updates.

15.12.8 Neuron K8 release [1.7.7.0]

Date: 01/20/2022

Summary

Minor updates

15.12.9 Neuron K8 release [1.7.3.0]

Date: 10/27/2021

Summary

Minor updates

15.12.10 [1.6.22.0]

Date: 08/30/2021

Summary

Minor updates.

15.12.11 [1.6.15.0]

Date: 08/06/2021

Summary

Minor updates.

15.12.12 [1.6.7.0]

Date: 07/26/2021

Summary

Minor internal enhancements.

15.12.13 [1.6.0.0]

Date: 07/02/2021

Summary

Minor internal enhancements.

15.12.14 [1.5.3.0]

Date: 05/01/2021

Summary

Minor internal enhancements.

15.12.15 [1.4.1.0]

Date: 01/30/2021

Summary

Minor internal enhancements.

15.12.16 [1.3.2.0]

Date: 12/23/2020

Summary

Minor internal enhancements.

15.12.17 [1.2.0.0]

Date: 11/17/2020

Summary

Minor internal enhancements.

15.12.18 [1.1.23.0]

Date: 10/22/2020

Summary

Support added for use with Neuron Runtime 1.1. More details in the Neuron Runtime release notes at neuron-runtime-release-notes.

15.12.19 [1.1.17.0]

Date: 09/22/2020

Summary

Minor internal enhancements.

15.12.20 [1.0.11000.0]

Date: 08/08/2020

Summary

First release of the Neuron K8 Scheduler extension.

Major New Features

- New scheduler extension is provided to ensure that kubelet is scheduling pods on inf1 with contiguous device ids. Additional details about the new scheduler are provided neuron-k8-scheduler-ext. including instructions on how to apply it.
  - NOTE: The scheduler is only required when using inf1.6xlarge and/or inf1.24xlarge
- With this release the device plugin now requires RBAC permission changes to get/patch NODE/POD objects. Please apply the k8s-neuron-device-plugin-rbac.yml before using the new device plugin.

Resolved Issues

- Scheduler is intended to address https://github.com/aws/aws-neuron-sdk/issues/110

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

SIXTEEN

DEVELOPER FLOWS

Neuron can be used in a wide selection of development flows. Each flow has its own starting point and requirements which are required to enable deep learning acceleration with AWS Neuron.

Deploy Containers with Neuron

Inference and Training

AWS EC2

Inference

This document is relevant for: `Inf1`

16.1 Compile with Framework API and Deploy on EC2 Inf1

Table of Contents

- *Description*
- *Setup Environment*
  - *1. Launch an Inf1 Instance*
  - *2. Set up a development environment*
    - *Enable PyTorch-Neuron*
    - *Enable TensorFlow-Neuron*
    - *Enable Apache MXNet (Incubating)*
  - *3. Set up Jupyter notebook*

16.1.1 Description

You can use a single inf1 instance as a development environment to compile and deploy Neuron models. In this developer flow, you provision an EC2 inf1 instance using a Deep Learning AMI (DLAMI) and execute the two steps of the development flow in the same instance. The DLAMI comes pre-packaged with the Neuron frameworks, compiler, and required runtimes to complete the flow. Development happens through Jupyter Notebooks or using a secure shell (ssh) connection in terminal. Follow the steps bellow to setup your environment.

Note: **Model compilation can be executed on a non-inf1 instance** for later deployment. Follow the same EC2 Developer Flow Setup using other instance families and leverage Amazon Simple Storage Service (S3) to share the compiled models between different instances.

16.1.2 Setup Environment

1. Launch an Inf1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
- When choosing an Amazon Machine Image (AMI) make sure to select Deep Learning AMI with Conda Options. Please note that Neuron Conda environments are supported only in Ubuntu 18 DLAMI and Amazon Linux2 DLAMI, Neuron Conda environments are not supported in Amazon Linux DLAMI.
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Note: You can also launch the instance from AWS CLI, please see AWS CLI commands to launch inf1 instances.

2. Set up a development environment

Enable PyTorch-Neuron

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

PyTorch 1.9.1

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

##################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
##################################################################################


# Activate PyTorch
source activate
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate
```

PyTorch 1.8.1

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
```

(continues on next page)

(continued from previous page)

```
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

#######################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
#######################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
```

(continues on next page)

(continued from previous page)

```
##################################################################################


# Activate PyTorch
source activate aws_neuron_pytorch_p36
```

Enable TensorFlow-Neuron

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

TensorFlow 2.5.1

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
```

(continues on next page)

(continued from previous page)

```
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

#######################################################
#   Only for Ubuntu 20 - Install Python3.7
#
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt-get install python3.7
#
#######################################################
# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt-get install -y python3.7-venv g++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
↪(Neuron TensorFlow)"
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo apt-get install tensorflow-model-server-neuronx --allow-change-held-packages -y
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

# Install Neuron TensorBoard
pip install tensorboard-plugin-neuron

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo yum install -y python3.7-venv gcc-c++
python3.7 -m venv tensorflow_venv
source tensorflow_venv/bin/activate
pip install -U pip


# Instal Jupyter notebook kernel
pip install ipykernel
python3.7 -m ipykernel install --user --name tensorflow_venv --display-name "Python␣
(Neuron TensorFlow)"
```

(continues on next page)

(continued from previous page)

```
pip install jupyter notebook
pip install environment_kernels


# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron TensorFlow
pip install tensorflow-neuron[cc] "protobuf"

# Optional: Install Neuron TensorFlow model server
sudo yum versionlock delete tensorflow-model-server-neuronx
sudo yum install tensorflow-model-server-neuronx -y
```

TensorFlow 1.15.5

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate TensorFlow
source activate aws_neuron_tensorflow_p36
```

Enable Apache MXNet (Incubating)

Important:

For successful installation or update to next releases (Neuron 1.20.0 and newer):

- Uninstall `aws-neuron-dkms` by running: `sudo apt remove aws-neuron-dkms` or `sudo yum remove aws-neuron-dkms`
- Install or upgrade to latest Neuron driver (`aws-neuron-dkms`) by following the "Setup Guide" instructions.

MXNet 1.8.0

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo apt-get update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################

# Install Neuron Tools
sudo apt-get install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Note: There is no DLAMI Conda environment for this framework version
#       Framework will be installed/updated inside a Python environment

# Update OS packages
sudo yum update -y

##########################################################################################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade to latest Neuron driver
##########################################################################################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################

# Install Neuron Tools
sudo yum install aws-neuronx-tools -y

export PATH=/opt/aws/neuron/bin:$PATH

# Activate MXNet
source activate aws_neuron_mxnet_p36

# Set Pip repository  to point to the Neuron repository
pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com

#Install Neuron MXNet
wget https://aws-mx-pypi.s3.us-west-2.amazonaws.com/1.8.0/aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install aws_mx-1.8.0.2-py2.py3-none-manylinux2014_x86_64.whl
pip install mx_neuron neuron-cc
```

MXNet 1.5.1

Ubuntu DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Install Neuron Driver
sudo apt-get install aws-neuronx-dkms --allow-change-held-packages -y

#######################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#######################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

Amazon Linux DLAMI

Note: For a successful installation or update, execute each line of the instructions below separately or copy the contents of the code block into a script file and source its contents.

```
# Neuron is pre-installed on Deep Learning AMI (DLAMI), latest DLAMI version may not␣
↪include latest Neuron versions
# To update to latest Neuron version, follow "Update to latest release" instruction on␣
↪Neuron documentation

##########################################################################################
↪#######################
# To install or update to Neuron versions 1.19.1 and newer from previous releases:
```

(continues on next page)

(continued from previous page)

```
# - DO NOT skip 'aws-neuron-dkms' install or upgrade step, you MUST install or upgrade␣
↪to latest Neuron driver
##########################################################################################
↪#######################

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Install Neuron Driver
sudo yum versionlock delete aws-neuronx-dkms
sudo yum install aws-neuronx-dkms -y

#################################################################################
# Warning: If Linux kernel is updated as a result of OS package update
#          Neuron driver (aws-neuron-dkms) should be re-installed after reboot
#################################################################################


# Activate MXNet
source activate aws_neuron_mxnet_p36
```

3. Set up Jupyter notebook

To develop from a Jupyter notebook see setup-jupyter-notebook-steps-troubleshooting

You can also run a Jupyter notebook as a script, first enable the ML framework Conda or Python environment of your choice and see running-jupyter-notebook-as-script for instructions.

This document is relevant for: `Inf1`

Training

This document is relevant for: `Trn1`

16.2 Train your model on EC2

Table of Contents

- *Description*
- *Setup Environment*
  - *1. Launch an Trn1 Instance*
  - *2. Set up a development environment*
    - *Enable PyTorch-Neuron*
  - *3. Set up Jupyter notebook*

16.2.1 Description

You can use a single Trn1 instance as a development environment to compile and train Neuron models. In this developer flow, you provision an EC2 Trn1 instance using a Deep Learning AMI (DLAMI) and execute the two steps of the development flow in the same instance. The DLAMI comes pre-packaged with the Neuron frameworks, compiler, and required runtimes to complete the flow. Development happens through Jupyter Notebooks or using a secure shell (ssh) connection in terminal. Follow the steps bellow to setup your environment.

16.2.2 Setup Environment

1. Launch an Trn1 Instance

- Please follow the instructions at launch an Amazon EC2 Instance to Launch an Trn1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Trn1 instances sizes and pricing see Trn1 web page.
- Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Ubuntu 18 AMI or Amazon Linux 2 AMI, you can also choose Ubuntu 18 or Amazon Linux 2 Deep Learning AMI (DLAMI)
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.
- After launching the instance, follow the instructions in Connect to your instance to connect to the instance

Important: AutoScalingGroups is currently not supported on Trn1 and will be added soon.

To launch a Trn1 cluster you can use AWS ParallelCluster, please see example.

Note: Neuron Driver installed on Deep Learning AMI (DLAMI) with Conda does not support Trn1.

If you want to use DLAMI with Conda, please make sure to uninstall `aws-neuron-dkms` and install `aws-neuronx-dkms` before using Neuron on DLAMI with Conda.

2. Set up a development environment

Enable PyTorch-Neuron

PyTorch 1.11.0

Ubuntu 20 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates
. /etc/os-release

sudo tee /etc/apt/sources.list.d/neuron.list > /dev/null <<EOF
deb https://apt.repos.neuron.amazonaws.com ${VERSION_CODENAME} main
EOF
wget -qO - https://apt.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-AWS-
↪NEURON.PUB | sudo apt-key add -

# Update OS packages
sudo apt-get update -y

# Install git
sudo apt-get install git -y


# Install OS headers
sudo apt-get install linux-headers-$(uname -r) -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo apt-get remove aws-neuron-dkms  -y
sudo apt-get remove aws-neuronx-dkms  -y
sudo apt-get remove aws-neuronx-oci-hook  -y
sudo apt-get remove aws-neuronx-runtime-lib -y
sudo apt-get remove aws-neuronx-collectives -y
sudo apt-get install aws-neuronx-dkms=2.* -y
sudo apt-get install aws-neuronx-oci-hook=2.* -y
sudo apt-get install aws-neuronx-runtime-lib=2.* -y
sudo apt-get install aws-neuronx-collectives=2.* -y

# Install EFA Driver(only required for multiinstance training)
```

(continues on next page)

(continued from previous page)

```
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import␣
↪aws-efa-installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig &&
↪ gpg --verify ./aws-efa-installer-latest.tar.gz.sig

tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo apt-get remove aws-neuron-tools  -y
sudo apt-get remove aws-neuronx-tools  -y
sudo apt-get install aws-neuronx-tools=2.* -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
sudo apt install python3.8-venv
python3.8 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
pip install -U pip

# Install wget, awscli
pip install wget
pip install awscli

# Install packages from  repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.
↪amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.*" "neuronx-cc==2.*"
```

Amazon Linux 2 AMI

Note:

- Instructions in this page only apply to setting up Neuron components on Linux host running Ubuntu or Amazon Linux AMI.
- When launching a Trn1, please adjust your primary EBS volume size to a minimum of 512GB.

```
# Configure Linux for Neuron repository updates

sudo tee /etc/yum.repos.d/neuron.repo > /dev/null <<EOF
[neuron]
```

(continues on next page)

(continued from previous page)

```
name=Neuron YUM Repository
baseurl=https://yum.repos.neuron.amazonaws.com
enabled=1
metadata_expire=0
EOF
sudo rpm --import https://yum.repos.neuron.amazonaws.com/GPG-PUB-KEY-AMAZON-
↪AWS-NEURON.PUB

# Install OS headers
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y

# Update OS packages
sudo yum update -y

# Install git
sudo yum install git -y

# Remove preinstalled packages and Install Neuron Driver and Runtime
sudo yum remove aws-neuron-dkms -y
sudo yum remove aws-neuronx-dkms -y
sudo yum remove aws-neuronx-oci-hook -y
sudo yum remove aws-neuronx-runtime-lib -y
sudo yum remove aws-neuronx-collectives -y
sudo yum install aws-neuronx-dkms-2.*  -y
sudo yum install aws-neuronx-oci-hook-2.*  -y
sudo yum install aws-neuronx-runtime-lib-2.*  -y
sudo yum install aws-neuronx-collectives-2.*  -y

# Install EFA Driver(only required for multiinstance training)
curl -O https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz
wget https://efa-installer.amazonaws.com/aws-efa-installer.key && gpg --import␣
↪aws-efa-installer.key
cat aws-efa-installer.key | gpg --fingerprint
wget https://efa-installer.amazonaws.com/aws-efa-installer-latest.tar.gz.sig &&
↪ gpg --verify ./aws-efa-installer-latest.tar.gz.sig
tar -xvf aws-efa-installer-latest.tar.gz
cd aws-efa-installer && sudo bash efa_installer.sh --yes
cd
sudo rm -rf aws-efa-installer-latest.tar.gz aws-efa-installer

# Remove pre-installed package and Install Neuron Tools
sudo yum remove aws-neuron-tools  -y
sudo yum remove aws-neuronx-tools  -y
sudo yum install aws-neuronx-tools-2.*  -y

export PATH=/opt/aws/neuron/bin:$PATH

# Install Python venv and activate Python virtual environment to install
# Neuron pip packages.
python3.7 -m venv aws_neuron_venv_pytorch
source aws_neuron_venv_pytorch/bin/activate
python -m pip install -U pip
```

(continues on next page)

(continued from previous page)

```
# Install wget, awscli
pip install wget
pip install awscli

# Install packages from repos
python -m pip config set global.extra-index-url "https://pip.repos.neuron.
↪amazonaws.com"

# Install Python packages - Transformers package is needed for BERT
python -m pip install torch-neuronx=="1.11.0.1.*" "neuronx-cc==2.*"
```

3. Set up Jupyter notebook

To develop from a Jupyter notebook see setup-jupyter-notebook-steps-troubleshooting

You can also run a Jupyter notebook as a script, first enable the ML framework Conda or Python environment of your choice and see running-jupyter-notebook-as-script for instructions.

This document is relevant for: `Trn1`

Amazon EKS

Inference

This document is relevant for: `Inf1`

16.3 Deploy Neuron Container on Elastic Kubernetes Service (EKS)

Table of Contents

- *Description*
- *Setup Environment*
  - *Self-managed Kubernetes*

16.3.1 Description

You can use the Neuron version of the AWS Deep Learning Containers to run inference on Amazon Elastic Kubernetes Service (EKS). In this developer flow, you set up an EKS cluster with Inf1 instances, create a Kubernetes manifest for your inference service and deploy it to your cluster. This developer flow assumes:

1. The model has already been compiled through *Compilation with Framework API on EC2 instance* or through *Compilation with Sagemaker Neo*.
2. You already set up your container to retrieve it from storage.

16.3.2 Setup Environment

1. **Install pre-requisits:** Follow these instruction to install or upgrade the *eksctl* command line utility on your local computer.

   Follow these instruction to install *kubectl* in the same computer. *kubectl* is a command line tool for working with Kubernetes clusters.

2. Follow the instructions in this EKS documentation link to set up AWS Inferentia on your EKS cluster. Using the YML deployment manifest shown in the same link, replace the *image* in the *containers* specification with the one you built using how-to-build-neuron-container above.

   Note: Before deploying your task definition to your EKS cluster, make sure to push the image to ECR. Refer to Pushing a Docker image for more information.

Self-managed Kubernetes

Please refer to tutorial-k8s-env-setup-for-neuron. In *Deploy a TensorFlow Resnet50 model as a Kubernetes service*, the container image referenced in the YML manifest is created using how-to-build-neuron-container.

This document is relevant for: `Inf1`

Training

Note: Amazon EKS support is coming soon.

Amazon ECS

Inference

This document is relevant for: `Inf1`

16.4 Deploy Neuron Container on Elastic Container Service (ECS)

Table of Contents

- *Description*
- *Setup Environment*

16.4.1 Description

You can use the Neuron version of the AWS Deep Learning Containers to run inference on Amazon Elastic Container Service (ECS). In this developer flow, you set up an ECS cluster with inf1 instances, create a task description for your inference service and deploy it to your cluster. This developer flow assumes:

1. The model has already been compiled through *Compilation with Framework API on EC2 instance* or through *Compilation with Sagemaker Neo*.

2. You already set up your container to retrieve it from storage.

16.4.2 Setup Environment

1. **Set up an Amazon ECS cluster:** Follow the instructions on Setting up Amazon ECS for Deep Learning Containers
2. **Define an Inference Task:** Use the instruction on the DLC Inference on ECS Tutorial to define a task and create a service for the appropriate framework.

   When creating tasks for inf1 instances on ECS, be aware of the considerations and requirements listed in Working with inference workloads on Amazon ECS.
3. Use the container image created using how-to-build-neuron-container as the `image` in your task definition.

Note: Before deploying your task definition to your ECS cluster, make sure to push the image to ECR. Refer to Pushing a Docker image for more information.

This document is relevant for: `Inf1`

Training

Note: Amazon ECS supports Trn1.

An example of how to train a model with Neuron using ECS is coming soon.

AWS Sagemaker

Inference

This document is relevant for: `Inf1`

16.5 Compile with Sagemaker Neo and Deploy on Sagemaker Hosting

Table of Contents

- *Description*
- *Setup Environment*

16.5.1 Description

You can use SageMaker Neo to compile models for deployment on SageMaker Hosting using ml.inf1 instances. In this developer flow, you provision a Sagemaker Notebook instance to train, compile and deploy your model using the SageMaker Python SDK. Follow the steps bellow to setup your environment.

16.5.2 Setup Environment

1. Create an Amazon SageMaker Notebook Instance:

   Follow the instructions in Get Started with Notebook Instances

   The Notebook instance created provides the required Python SDK for training, compiling and deploying models with Amazon SageMaker.

2. Compile a model using the Amazon SageMaker SDK:

   Refer to Supported Instances Types and Frameworks for information on the framework versions currently supported by Amazon SageMaker Neo on AWS Inferentia.

   More information about compiling and deploying models with Amazon SageMaker Neo can be found on Use Neo to Compile a Model

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

16.6 Bring Your Own Neuron Container to Sagemaker Hosting

Table of Contents

- *Description*
- *Setup Environment*

16.6.1 Description

You can use a SageMaker Notebook or an EC2 instance to compile models and build your own containers for deployment on SageMaker Hosting using ml.inf1 instances. In this developer flow, you provision a Sagemaker Notebook or an EC2 instance to train and compile your model to Inferentia. Then you deploy your model to SageMaker Hosting using the SageMaker Python SDK. Follow the steps bellow to setup your environment. Once your environment is set you'll be able to follow the *BYOC HuggingFace pretrained BERT container to Sagemaker Tutorial* .

16.6.2 Setup Environment

1. **Create a Compilation Instance:** If using an **EC2 instance for compilation** you can use an Inf1 instance to compile and test a model. Follow these steps to launch an Inf1 instance:

   - Please follow the instructions at launch an Amazon EC2 Instance to Launch an Inf1 instance, when choosing the instance type at the EC2 console. Please make sure to select the correct instance type. To get more information about Inf1 instances sizes and pricing see Inf1 web page.
   - Select your Amazon Machine Image (AMI) of choice, please note that Neuron support Ubuntu 18 AMI or Amazon Linux 2 AMI, you can also choose Ubuntu 18 or Amazon Linux 2 Deep Learning AMI (DLAMI)
   - After launching the instance, follow the instructions in Connect to your instance to connect to the instance

   If using an **SageMaker Notebook for compilation**, follow the instructions in Get Started with Notebook Instances to provision the environment.

   It is recommended that you start with an ml.c5.4xlarge instance for the compilation. Also, increase the volume size of you SageMaker notebook instance, to accomodate the models and containers built locally. A volume of 10GB is sufficient.

   > **Note:** To compile the model in the SageMaker Notebook instance, you'll need to update the conda environments to include the Neuron Compiler and Neuron Framework Extensions. Follow the installation guide on the section how-to-update-to-latest-Neuron-Conda-Env to update the environments.

2. **Set up the environment to compile a model, build your own container and deploy:** To compile your model on EC2 or SageMaker Notebook, follow the *Set up a development environment* section on the EC2 *Setup Environment* documentation.

   Refer to Adapting Your Own Inference Container documentation for information on how to bring your own containers to SageMaker Hosting.

   Make sure to add the **AmazonEC2ContainerRegistryPowerUser** role to your IAM role ARN, so you're able to build and push containers from your SageMaker Notebook instance.

   Note: The container image can be created using how-to-build-neuron-container.

This document is relevant for: `Inf1`

Training

This document is relevant for: `Trn1`

16.7 Train your model on SageMaker

Table of Contents

- *Description*
- *Setup environment*

16.7.1 Description

SageMaker Training helps you manage cloud computing resources in Amazon EC2, data storage services such as S3, EFS, and FSx, and security management services such as IAM and VPC. SageMaker Training provides you a complete end-to-end experience of training classical ML and state-of-the-art DL models.

You can use SageMaker to train models using Trn1 instances (ml.trn1 instance types). In this developer flow, you provision a SageMaker Notebook instance or SageMaker Studio to train your model using the SageMaker Python SDK.

The Amazon SageMaker Python SDK lets you launch training jobs in just a few lines of code with ease. As shown in the below diagram Amazon SageMaker launches Trn1 instances, copies both data and code onto the instance. It then runs the training script to generate model artifacts. The trained model artifacts are then uploaded to S3 and finally SageMaker will terminate the provisioned instances. In order to speed up the training process for successive runs you can copy the Neuron Persistent Cache to S3 and then copied by future training jobs as they will leverage the cached artifacts. (See Hugging Face fine tuning BERT base model on Amazon SageMaker Tutorial for an example on how to reuse the compiled cache.)

16.7.2 Setup environment

1. Create an Amazon SageMaker Notebook Instance

   Follow the instructions in Get Started with Notebook Instances or Use Amazon SageMaker Studio Notebooks. The Notebook instance provides the required Python SDK for training models with Amazon SageMaker. Please make sure SageMaker Python SDK version is 2.116.0 or later.

2. Train a model using the Amazon SageMaker SDK

   Follow the instructions in Distributed Training with PyTorch Neuron on Trn1 instances. You'll be able to follow the Hugging Face fine tuning BERT base model on Amazon SageMaker Tutorial.

   Note: SageMaker support for EC2 Trn1 instance is currently available only for PyTorch Estimator. HuggingFace Estimator will be available in future release.

This document is relevant for: `Trn1`

AWS ParallelCluster

Training

This document is relevant for: `Trn1`

16.8 Train your model on ParallelCluster

Table of Contents

- *Description*
- *Setup environment*

16.8.1 Description

This document explains how to use AWS ParallelCluster to build HPC compute environment that uses Trn1 compute nodes to run your distributed ML training job. Once the nodes are launched, we will run a training task to confirm that the nodes are working, and use slurm commands to check the job status. In this tutorial, we will use AWS *pcluster* command to run a yaml file in order to generate the cluster. As an example, we are going to launch multiple Trn1.32xl nodes in our cluster.

We are going to set up our ParallelCluster infrastructure as below:

As shown in the figure above, inside a VPC, there are two subnets, a public and a private ones. Head Node resides in the public subnet, while the compute fleet (in this case, trn1 instances) are in the private subnet. A Network Address Translation (NAT) gateway is also needed in order for nodes in the private subnet to connect to clients outside the VPC. In the next section, we are going to describe how to set up all the necessary infrastructure for trn1 ParallelCluster.

16.8.2 Setup environment

1. Install prerequisite infrastructure:

Follow these setup instructions to install VPC and all the necessary components for ParallelCluster.

2. Create and launch ParallelCluster

Follow these creating cluster instructions to launch ParallelCluster in the VPC.

1. Launch training job

Follow these running training instructions to submit a model training script as a slurm job.

This document is relevant for: `Trn1`

Inference

Note: AWS ParallelCluster support is coming soon.

AWS Batch

Inference

Note: AWS Batch supports Inf1.

An example of how to deploy a model with Neuron using Batch is coming soon.

Training

Note: AWS Batch support is coming soon.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

SEVENTEEN

NEURON ARCHITECTURE

The Neuron Archtecture provides insights into Neuron enabled instances system, software and chip capabilities. The EC2 Trn1 and Inf1 instance architecture provides an overview of the EC2 instances powered by the Inferentia and Trainium chips (Neuron Devices), and the corresponding system features like inbox and network connectivity, memory hierarchy, and NeuronCores versions and capabilities. The Neuron model architecture fit provides insights to what is the best match between deep-learning model architectures and the NeuronCore version.

EC2 Inf1 Architecture EC2 Trn1 Architecture AWS NeuronCore Architecture Neuron Model Architecture Fit Guidelines Neuron Glossary

This document is relevant for: `Inf1`

17.1 AWS Inf1 Architecture

In this page, we provide an architectural overview of the AWS Inf1 instances, and the corresponding *Inferentia* NeuronDevices that power them (*Inferentia* devices from here on).

Table of contents

- *Inf1 Architecture*
- *Inferentia Architecture*

17.1.1 Inf1 Architecture

The EC2 Inf1 instance is powered by 16 *Inferentia* devices, and allows customers to choose between four instances sizes:

Instance size	# of Inferentia devices	vCPUs	Host Memory (GiB)	FP16/BF16 TFLOPS	INT8 TOPS	Device Memory (GiB)	Device Memory Bandwidth (GiB/sec)	NeuronLink-v1 device-to-device bandwidth (GiB/sec/device)	EFA bandwidth (Gbps)
Inf1.xlarge	1	4	8	64	128	8	50	N/A	up-to 25
Inf1.2xlarge	1	8	16	64	128	8	50	N/A	up-to 25
Inf1.6xlarge	4	24	48	256	512	32	200	32	25
Inf1.24xlarge	16	96	192	1024	2048	128	800	32	100

Inf1 offers a direct device-to-device interconnect called NeuronLink-v1, which enables co-optimizing latency and throughput via the *Neuron Core Pipeline <neuroncore-pipeline>* technology.

17.1.2 Inferentia Architecture

At the heart of the Inf1 instance are 16 Inferentia devices, as depicted below:

Each Inferentia device consists of:

- **Compute:**
  - 4x *NeuronCore-v1* cores, delivering 128 INT8 TOPS and 64 FP16/BF16 TFLOPS

- **Device Memory:**
  - 8GB of device DRAM memory (for storing parameters and intermediate state), with 50 GB/sec of bandwidth
- **NeuronLink:**
  - Enables co-optimization of latency and throughput via the *Neuron Core Pipeline* technology

This document is relevant for: `Inf1`

This document is relevant for: `Trn1`

17.2 AWS Trn1 Architecture

In this page, we provide an architectural overview of the AWS Trn1 instances, and the corresponding *Trainium* NeuronDevices that power them (Trainium devices from here on).

Table of contents

- *Trn1 Architecture*
- *Trainium Architecture*

17.2.1 Trn1 Architecture

The EC2 Trn1 instance is powered by up to 16 *Trainium* devices, and allows customers to choose between 2 instance sizes:

Instance size	# of Trainium devices	vCPUs	Host Memory (GiB)	FP8/FP16/BF16/TF32 TFLOPS	FP32 TFLOPS	Device Memory (GiB)	Device Memory Bandwidth (GiB/sec)	NeuronLink-v2 device-to-device	EFA bandwidth (Gbps)
Trn1.2xlarge	1	8	32	210	52.5	32	820	N/A	up-to 25
Trn1.32xlarge	16	128	512	3,360	840	512	13,120	Yes	800

The Trn1.2xlarge instance size allows customers to train their models on a single Trainium device, which is useful for small model training, as well as model experimentation. The Trn1.32xlarge instance size comes with a high-bandwidth and low-latency NeuronLink-v2 device-to-device interconnect, which utilizes a 4D-HyperCube topology. This is useful for collective-communication between the Trainium devices during scale-out training, as well as for pooling the memory capacity of all Trainium devices, making it directly addressable from each one of the devices.

In a Trn1 server, the Trainium devices are connected in a 2D Torus topology, as depicted below:

The Trn1 instances are also available in an EC2 UltraCluster, which enables customers to scale Trn1 instances to over 30,000 Trainium devices, and leverage the AWS-designed non-blocking petabit-scale EFA networking infrastructure.

17.2.2 Trainium Architecture

At the heart of the Trn1 instance are Trainium devices, the second generation purpose-built Machine Learning accelerator from AWS. The Trainium device architecture is depicted below:

Each Trainium device consists of:

- **Compute:**
  - 2x *NeuronCore-v2* cores, delivering 420 INT8 TOPS, 210 FP16/BF16/cFP8/TF32 TFLOPS, and 52.5 FP32 TFLOPS.
- **Device Memory:**
  - 32GB of device memory (for storing model state), with 820 GB/sec of bandwidth.
- **Data movement:**
  - 1 TB/sec of DMA bandwidth, with inline memory compression/decompression.
- **NeuronLink:**
  - NeuronLink-v2 for device-to-device interconnect enables efficient scale-out training, as well as memory pooling between the different Trainium devices.
- **Programmability:**
  - Trainium supports dynamic shapes and control flow, via ISA extensions of NeuronCore-v2. In addition, Trainium also allows for user-programmable *rounding mode* (Round Nearest Even Stochastic Rounding), and custom-operators via the deeply embedded GPSIMD Engine.

This document is relevant for: `Trn1`

This document is relevant for: `Inf1`, `Trn1`

17.3 AWS NeuronCore Architecture

In this page, we provide an architectural overview of the AWS NeuronCore architecture across generations.

Table of contents

- *NeuronCore-v1 Architecture*
- *NeuronCore-v2 Architecture*

17.3.1 NeuronCore-v1 Architecture

NeuronCore-v1 is the first generation of the NeuronCore engine, powering the Inferentia NeuronDevices. Each NeuronCore-v1 is a fully-independent heterogenous compute-unit, with 3 main engines (Tensor/Vector/Scalar Engines), and on-chip software-managed SRAM memory, for maximizing data locality (compiler managed, for maximum data locality and optimized data prefetch).

The ScalarEngine is optimized for scalar-computations, in which every element of the output is dependent on one element of the input, e.g. non-linearities like GELU, SIMGOID or EXP. The ScalarEngine is highly parallelized, and can process 512 floating point operations per cycle. It can handle various data-types, including FP16, BF16, FP32, INT8, INT16 and INT32.

The VectorEngine is optimized for vector-computations, in which every element of the output is dependent on multiple input elements. Examples include 'axpy' operations (Z=aX+Y), Layer Normalization, Pooling operations, and many more. The VectorEngine is also highly parallelized, and can perform 256 floating point operations per cycle. It can handle various data-types, including FP16, BF16, FP32, INT8, INT16 and INT32.

The TensorEngine is based on a power-optimized systolic-array which is highly optimized for tensor computations (e.g. GEMM, CONV, Reshape, Transpose), and supports mixed-precision computations (FP16/BF16/INT8 inputs, FP32/INT32 outputs). Each NeuronCore-v1 TensorEngine delivers 16 TFLOPS of FP16/BF16 tensor computations.

17.3.2 NeuronCore-v2 Architecture

NeuronCore-v2 is the second generation of the NeuronCore engine, powering the Trainium NeuronDevices. Each NeuronCore-v2 is a fully-independent heterogenous compute-unit, with 4 main engines (Tensor/Vector/Scalar/GPSIMD Engines), and on-chip software-managed SRAM memory, for maximizing data locality (compiler managed, for maximum data locality and optimized data prefetch).

Just like in NeuronCore-v1, The ScalarEngine is optimized for scalar-computations, in which every element of the output is dependent on one element of the input. The ScalarEngine is highly parallelized, and can perform 1,600 floating point operations per cycle (3x speedup relative to NeuronCore-v1). The NeuronCore-v2 ScalarEngine can handle various data-types, including cFP8, FP16, BF16, TF32, FP32, INT8, INT16 and INT32.

The VectorEngine is optimized for vector-computations, in which every element of the output is dependent on multiple input elements. Examples include 'axpy' operations (Z=aX+Y), Layer Normalization, Pooling operations, and many more. The VectorEngine is also highly parallelized, and can perform 2,500 floating points operations per cycle (10x speedup vs NeuronCore-v1). The NeuronCore-v2 ScalarEngine can handle various data-types, including cFP8, FP16, BF16, TF32, FP32, INT8, INT16 and INT32.

The TensorEngine is based on a power-optimized systolic-array which is highly optimized for tensor computations (e.g. GEMM, CONV, Reshape, Transpose), and supports mixed-precision computations (cFP8 / FP16 / BF16 / TF32 / FP32 / INT8 inputs, FP32 / INT32 outputs). Each NeuronCore-v2 TensorEngine delivers over 100 TFLOPS of FP16/BF16 tensor computations (a 6x speedup from NeuronCore-v1).

NeuronCore-v2 also introduces a new engine, called GPSIMD-Engine. This engine consists of 8 fully programmable 512-bit wide general-purpose processors, which can execute straight-line C-code, and have direct access to the other NeuronCore-v2 engines, as well as the embedded on-chip SRAM memory. With these cores, customers can implement custom-operators and execute them directly on the NeuronCore engines.

NeuronCore-v2 also adds support for control-flow, dynamic-shapes, and programmable *rounding mode* (RNE & Stochastic-rounding).

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

17.4 Model Architecture Fit Guidelines

Table of contents

- *Introduction*
- *Model Support Overview*
  - *AWS Trainium (NeuronCore v2)*
  - *AWS Inferentia (NeuronCore v1)*
- *Natural Language Processing (NLP) Models with Transformers*
  - *Encoder models, or autoencoding models with Transformers*
  - *Decoder models, or autoregressive models with Transformers*
  - *Encoder-decoder models, or sequence-to-sequence models with Transformers*
- *Computer Vision Models*
  - *Convolutional Neural Network (CNN) based models*
  - *Region-based CNN (R-CNN) models*
- *Models with Long Short-Term Memory (LSTM) networks*
  - *Trainium*
  - *Inferentia*
- *Diffusion Models*
  - *Trainium*
  - *Inferentia*
- *Known Issues on Inferentia (NeuronCore v1)*
  - *Support of large models (impacts* torch-neuron *and* tensorflow-neuron *(TF1.x))*

17.4.1 Introduction

AWS Neuron SDK enables you to train and deploy a wide range of deep learning models on AWS Inferentia and Trainium. Inferentia is the first generation NeuronDevice optimized for inference, and Trainium is the second generation that adds support for model training.

This document describes what types of deep learning model architectures are a good fit for Inferentia and Trainium.

17.4.2 Model Support Overview

AWS Trainium (NeuronCore v2)

Last update - 10/10/2022

Model Family/ Neural Network Architecture	Category	Hardware Architecture	PyTorch Neuron (`torch-neuronx`)	TensorFlow Neuron (`tensorflow-neuronx`)
Transformers Encoders	NLP	Good Fit	Supported	*Roadmap Item*
Transformers Decoders	NLP	Good Fit	Supported	*Roadmap Item*
Transformers Encoder-Decoder (Sequence-to-sequence)	NLP	Good Fit	*Roadmap Item*	*Roadmap Item*
LSTMs	NLP and Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*
Vision Transformers	Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*
Diffusion models	Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*
Convolutional Neural Network (CNN) models	Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*
R-CNNs	Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*

Note: Supported means that at least a single model of the model family or the neural-network architecture already enabled.

AWS Inferentia (NeuronCore v1)

Last update - 10/10/2022

Model Family/ Neural Network Architecture	Category	Hardware Architecture	PyTorch Neuron (`torch-neuron`)	TensorFlow Neuron (`tensorflow-neuron` (`TF 1.x`))	TensorFlow Neuron (`tensorflow-neuron` (`TF 2.x`))
Transformers Encoders	NLP	Good Fit	Supported	Supported	Supported
Transformers Decoders	NLP	Not a Good Fit	NA	NA	NA
Transformers Encoder-Decoder (Sequence-to-sequence)	NLP	Not a Good Fit	NA	NA	NA
LSTMs	NLP and Computer Vision	Good Fit	Supported	NA	NA
Vision Transformers	Computer Vision	Good Fit	*Roadmap Item*	*Roadmap Item*	*Roadmap Item*
Diffusion models	Computer Vision	Good Fit	*Roadmap Item*	NA	NA
Convolutional Neural Network (CNN) models	Computer Vision	Good Fit	Supported	Supported	*Roadmap Item*
R-CNNs	Computer Vision	Supported with limitations	Supported with limitations	NA	NA

Note: Supported means that at least a single model of the model family or the neural-network architecture already enabled.

17.4.3 Natural Language Processing (NLP) Models with Transformers

Encoder models, or autoencoding models with Transformers

Autoencoding models use only the encoder part of the Transformer architecture. Representatives of this family include models like **BERT, distilBERT, XLM-BERT, Roberta, BioBert**, etc. Since the encoding process in these models can be parallelized, you can expect these models to run well both on Inferentia and Trainium.

Trainium

- **Architecture Fit** - Autoencoding models are a good fit for Trainium.
- **Neuron Support** - Neuron SDK supports Autoencoding models training and evaluation on Trainium. Please refer to the Trainium tutorials for more details.

Inferentia

- **Architecture Fit** - Autoencoding models are a good fit for Inferentia.
- **Neuron Support** - Neuron SDK support running Autoencoding models for inference on Inferentia. Please see benchmark results of these models. To get started with NLP models you can refer to Neuron PyTorch, TensorFlow and MXNet NLP tutorials.

Decoder models, or autoregressive models with Transformers

Autoregressive models keep only the decoder part of the Transformer architecture. Representatives of this family include models like **GPT-3, GPT-2**, etc.

Trainium

- **Architecture Fit** - Autoregressive models are a good fit for Trainium.
- **Neuron Support** - Neuron SDK supports Autoregressive models training and evaluation on Trainium. Please refer to the Trainium tutorials for more details.

Inferentia

- **Architecture Fit** - Autoregressive models are not a good fit for Inferentia. Usually the decoder part in these models is the most significant performance bottleneck since it must be executed once per output token, causing frequent access to the memory. Due to this these models typically experience the best performance only when the decoder maximum sequence length is short (e.g., 128).
- **Neuron Support** - Neuron SDK does not support Autoregressive models inference on Inferentia.

Encoder-decoder models, or sequence-to-sequence models with Transformers

Sequence-to-sequence models use both of encoder and decoder of the Transformer architecture. Representatives of this family include models like **T5, Bart, Marian MT**, etc.

Trainium

- **Architecture Fit** - Sequence-to-sequence models are a good fit for Trainium.
- **Neuron Support** - Sequence-to-sequence models are not supported on Trainium as of the latest Neuron release, please track the *Neuron Roadmap* for details.

Inferentia

- **Architecture Fit** - Sequence-to-sequence models are not a good fit for Inferentia. Like decoder models explained above, usually the decoder part in these sequence-to-sequence models is the most significant performance bottleneck since it must be executed once per output token, causing frequent access to the memory. Due to this, even when you enabled the models to run on Inferentia with wrapping the decoder part, these models typically experience the best performance only when the decoder maximum sequence length is short (e.g., 128).
- **Neuron Support** - Neuron SDK does not support sequence-to-sequence models inference on Inferentia out of the box. However, you can run a model with defining wrappers around the encoder and decoder portions of it. For example, please refer to *MarianMT tutorial* on Inferentia for more details.

17.4.4 Computer Vision Models

Convolutional Neural Network (CNN) based models

CNN based models are used for applications in image classification and object detection. Representatives of this family include models like **ResNet, ResNext, VGG, YOLO, SSD**, etc.

Trainium

- **Architecture Fit** - CNN based models are a good fit for Trainium.
- **Neuron Support** - CNN based models are not supported on Trainium as of the latest Neuron release, please track the *Neuron Roadmap* for details.

Inferentia

- **Architecture Fit** - CNN based models are a good fit for Inferentia.
- **Neuron Support** - Neuron SDK supports CNN based models inference on Inferentia. Please see the benchmark results of these models. To get started with these models you can refer to Neuron PyTorch, TensorFlow and MXNet tutorials.

Region-based CNN (R-CNN) models

Region-based CNNs (R-CNNs) models are commonly used for object detection and image segmentation tasks. Popular variants of the the R-CNN model include R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN.

Trainium

- **Architecture Fit** - R-CNN models are a good fit for Trainium.
- **Neuron Support** - R-CNN models are not supported on Trainium as of the latest Neuron release, please track the *Neuron Roadmap* for details.

Inferentia

- **Architecture Fit** - R-CNN models can have a few limitations and considerations on Inferentia: **RoI Align operators**: At this time, RoI Align operators typically cannot run efficiently on NeuronCore v1. As a result, RoI Align operators are mapped directly to CPU during compilation. R-CNN models that predict a low number of bounding boxes (<100) experience the best performance on Inferentia. **Large ResNet backbone**: R-CNNs that have a large ResNet backbone (such as ResNet-50 or ResNet-101) experience the greatest performance improvement on Inferentia because a larger portion of the R-CNN compute is accelerated.
- **Neuron Support** - Torch models must be traceable using `torch.jit.trace()` for compilation on Inferentia. Most Detectron2-based R-CNNs are not jit traceable by default, so they cannot readily be compiled for optimized inference on Inferentia. The *Running R-CNNs on Inf1* application note demonstrates how to compile and improve the performance of R-CNN models on Inferentia. It also provides an end-to-end example of running a Detectron2 R-CNN on Inferentia.

17.4.5 Models with Long Short-Term Memory (LSTM) networks

LSTMs use an internal state to process sequential data. LSTMs are commonly used to model temporal sequences of data in language processing and computer vision applications.

Trainium

- **Architecture Fit** - Models with LSTM networks are a good fit for Trainium.
- **Neuron Support** - Models with LSTM networks are not supported on Trainium as of the latest Neuron release. Please track the *Neuron Roadmap* for details.

Inferentia

- **Architecture Fit** - Models with LSTM cells are a good fit for Inferentia.
- **Neuron Support** - Models with LSTM networks are supported on Inferentia, please see *Developer Guide - PyTorch Neuron (torch-neuron) LSTM Support*.

17.4.6 Diffusion Models

Trainium

- **Architecture Fit** - Diffusion models are a good fit for Trainium.
- **Neuron Support** - Diffusion models are not supported on Trainium as of the latest Neuron release. Please track the *Neuron Roadmap* for details.

Inferentia

- **Architecture Fit** - Diffusion models are a good fit for Inferentia.
- **Neuron Support** - Diffusion models are not supported on Inferentia as of the latest Neuron release. Please track the *Neuron Roadmap* for details.

17.4.7 Known Issues on Inferentia (NeuronCore v1)

Support of large models (impacts *torch-neuron* and *tensorflow-neuron* (TF1.x))

During compilation on Inferentia (NeuronCore v1), `torch-neuron` and `tensorflow-neuron (TF1.x)` export a protobuf that contains the model's graph structure and weights. This causes an issue when the total size of the model's weights exceeds the 2GB limitation of protobufs. As a result, customers who want to run large models such as **RegNet**, **Stable Diffusion**, and **t5-11b** might run into protobuf errors during compilation.

This is a known issue related to the compilation process, not a hardware-dependent issue. Allowing large models like this to be compiled for inference on Inferentia (NeuronCore v1) is a feature that we intend to address in a future release. Please track the *Neuron Roadmap* for details.

Note: Neuron release 2.5.0 added Experimental support for tracing models larger than 2GB *in `tensorflow-neuron (TF2.x)`*, please see `extract-weights` flag in *TensorFlow 2.x (tensorflow-neuron) Tracing API*

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

17.5 Neuron Glossary

Table of contents

- *Terms*
  - *Neuron Devices (Accelerated Machine Learning chips)*
  - *Neuron powered Instances*
  - *NeuronCore terms*
- *Abbreviations*

17.5.1 Terms

Neuron Devices (Accelerated Machine Learning chips)

Term	Description
Inferentia	AWS first generation accelerated machine learning chip supporting inference only
Trainium	AWS second generation accelerated machine learning chip supporting training and inference
Neuron Device	Accelerated machine learning chip (e.g. Inferentia or Trainium)

Neuron powered Instances

Term	Description
Inf1	Inferentia powered accelerated compute EC2 instance
Trn1	Trainium powered accelerated compute EC2 instance

NeuronCore terms

Term	Description
NeuronCore	The machine learning compute cores within Inferentia/Trainium
NeuronCore-v1	Neuron Core withing Inferentia
NeuronCore-v2	Neuron Core withing Trainium
Tensor Engine	2D systolic array (within the NeuronCore), used for matrix computations
Scalar Engine	A scalar-engine within each NeuronCore, which can accelerate element-wise operations (e.g. GELU, ReLU, reciprocal, etc)
Vector Engine	A vector-engine with each NeuronCore, which can accelerate spatial operations (e.g. layerNorm, TopK, pooling, etc)
GPSIMD Engine	Embedded General Purpose SIMD cores, within each NeuronCore, to accelerate custom-operators
Sync Engine	The SP engine, which is integrated inside NeuronCore. Used for synchronization and DMA triggering.
Collective Communication Engine	Dedicated engine for collective communication, allows for overlapping computation and communication
NeuronLink	Interconnect between NeuronCores
NeuronLink-v1	Interconnect between NeuronCores in Inferentia device
NeuronLink-v2	Interconnect between NeuronCores in Trainium device

17.5.2 Abbreviations

Abbreviation	Description
NC	Neuron Core
NeuronCore	Neuron Core
ND	Neuron Device
NeuronDevice	Neuron Device
TensEng	Tensor Engine
ScalEng	Scalar Engine
VecEng	Vector Engine
SyncEng	Sync Engine
CCE	Collective Communication Engine
FP32	Float32
TF32	TensorFloat32
FP16	Float16
BF16	Bfloat16
cFP8	Configurable Float8
RNE	Round Nearest Even
SR	Stochastic Rounding
CustomOps	Custom Operators
RT	Neuron Runtime
DP	Data Parallel
DPr	Data Parallel degree
TP	Tensor Parallel
TPr	Tensor Parallel degree

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

EIGHTEEN

NEURON FEATURES

Neuron features provide insights into Neuron capabilities that enable high-performance and improve usability of developing and deploying deep learning acceleration on top of Inferentia and Trainium based instances.

Data Types Neuron Rounding Modes Neuron Batching NeuronCore Pipeline Neuron Persistent Cache Neuron Collective Communication Neuron Control Flow Neuron Custom C++ Operators Neuron Dynamic Shapes

This document is relevant for: `Inf1`, `Trn1`

18.1 Data Types

Table of contents

- *Introduction*
- *NeuronCore v1 Data Types*
  - *Neuron Data-Types*
- *NeuronCore v2 Data Types*
  - *Model Type Conversion*
  - *NeuronCore v2 Rounding Modes*

18.1.1 Introduction

Inferentia and Trainium NeuronDevices include different NeuronCore versions, which support differnt data-types. This section describes what data-types are supported in each NeuronCore version, for details about NeuronCore versions see neuron_hw_arch.

18.1.2 NeuronCore v1 Data Types

Neuron Data-Types

Neuron enables developers to choose from multiple data-types. The supported data-types are FP32, FP16, and BF16. Developers can train their models on their platform of choice (e.g. EC2 P3 instances), and then easily move their trained models to EC2 Inf1 for execution.

FP16/BF16 models

Models natively trained in FP16/BF16 will be executed in their trained data-types. This is a straightforward migration from the training platform to Inf1.

FP32 models

Neuron SDK supports **automatic model conversion** from FP32 to BF16 by default. This capability allows developers to train their models using FP32 format for the highest accuracy, and achieve performance benefits without having to worry about low-precision training (e.g. no need for loss-scaling during training). ML models are typically robust to FP32 to BF16 conversion, with minimal to no impact on accuracy. The conversion accuracy is model dependent; therefore, users are encouraged to benchmark the accuracy of the auto-converted model against the original FP32 trained model.

When the compiler is supplied with an unmodified FP32 model input it will automatically compile the model to run as BF16 on Inferentia. During inference the FP32 input data will be auto-converted internally by Inferentia to BF16 and the output will be converted back to FP32 data-type. For explicit FP16 inferencing, either use an FP16 trained model, or use an external tool (like AMP) to make the explicit conversions.

18.1.3 NeuronCore v2 Data Types

The NeuronCore v2 supports the following data types:

- 32 and 16-bit Floating Point (FP32 / FP16)
- TensorFloat-32 (TF32)
- Brain Floating Point (BFloat16)
- Unsigned 8-bit integer (UINT8)

Note: Neuron Compiler support for UINT8 is planned for a future Neuron SDK release. See Neuron Compiler: Enable Neuron INT8 support for details.

The layout for these is as follows:

Model Type Conversion

The Neuron SDK supports automatic model conversion from FP32 to BF16 by default. This capability allows developers to train their models using FP32 format for the highest accuracy, and then achieve run-time performance benefits without having to worry about low-precision training (e.g. no need for loss-scaling during training). ML models are typically robust to FP32 to BF16 conversion, with minimal to no impact on accuracy. Since conversion accuracy is model dependent, users are encouraged to benchmark the accuracy of the auto-converted model against the original FP32 trained model.

See *Mixed Precision and Performance-accuracy Tuning for Training* for more details on supported data types and their properties.

The Neuron compiler offers the `--auto-cast` and `--auto-cast-type` options to specify automatic casting of FP32 tensors to other data types to address performance and accuracy tradeoffs. See the *Neuron Compiler CLI Reference Guide* for a description of these options.

NeuronCore v2 Rounding Modes

Because floating point values are represented by a finite number of bits, they cannot represent all real numbers accurately. Floating point calculations that exceed their defined data type size are rounded. The NeuronCore v2 performs a Round-to-Nearest (RNE) algorithm with ties to Even by default. It also provides a new Stochastic Rounding mode. When Stochastic Rounding is enabled, the hardware will round the floating point value up or down using a proportional probability. This could lead to improved model convergence. Use the environment variable NEURON_RT_STOCHASTIC_ROUNDING_EN to select a rounding mode.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Trn1`

18.2 Neuron Rounding Modes

Table of contents

- *Round Nearest, ties to Even (RNE)*
- *Stochastic Rounding (SR)*
- *Quick Tests*

18.2.1 Round Nearest, ties to Even (RNE)

When the exact result of a floating point operation cannot be exactly represented as a floating point value, it must be rounded. The IEEE 754-2008 standard defines the default rounding mode to be 'Round Nearest, ties to Even' (RNE for short). Under this scheme, numbers are rounded to the nearest representable value, and in case of a 'tie' (i.e. the number is exactly between the two nearest representable values) numbers will be rounded to the nearest even number.

All NeuronCore generations support the RNE rounding scheme, which is the most commonly used rounding scheme for Machine Learning workloads. Below is an illustration of the RNE rounding scheme:

18.2.2 Stochastic Rounding (SR)

One downside of the RNE rounding scheme (and other rounding schemes described in the IEEE 754-2008 standard), is that when adding floating point values of significantly different magnitudes, rounding can squash small values and prevent them from accumulating over time.

To improve this, starting from the second generation of the NeuronCore (NeuronCore-v2), customers can choose between the RNE rounding scheme described above, and a second rounding scheme called 'Stochastic Rounding' (SR for short). Stochastic rounding prevents the computation precision-loss described above, by performing the rounding operations in a probabilistic manner, according to the relative distance from the two nearest representable values, as illustrated below:

By performing the rounding in a probabilistic manner, this scheme allows for small increments to accumulate over time, even when added to numbers of significantly higher magnitude, which leads to more precise results when performing large floating point computations (as done for machine learning).

18.2.3 Quick Tests

As an example, we examine the code-snippet below:

```
import torch
import torch_xla
import torch_xla.core.xla_model as xm
device = xm.xla_device()

a = torch.tensor(1024.0).half().to(device)

for i in range(2048) :
   a = (a + 0.5)
   xm.mark_step()

print(a)
```

This code shows that rounding can significantly impact the calculation's precision over time. To use standard RNE rounding, use the environment variable `NEURON_RT_STOCHASTIC_ROUNDING_EN=0`. To enable stochastic rounding, use the environment variable `NEURON_RT_STOCHASTIC_ROUNDING_EN=1`.

NOTE: Stochastic rounding mode is enabled by default in PyTorch-Neuron when XLA_USE_BF16=1.

The first test continues to show 1024 due to RNE rounding after each addition, and the second test shows result that is mostly in line with expectation.

```
$ NEURON_RT_STOCHASTIC_ROUNDING_EN=0 python3 rounding_mode_test.py

tensor(1024., device='xla:1', dtype=torch.float16)

$ NEURON_RT_STOCHASTIC_ROUNDING_EN=1 python3 rounding_mode_test.py

tensor(2056., device='xla:1', dtype=torch.float16)
```

This document is relevant for: `Trn1`

This document is relevant for: `Inf1`

18.3 Neuron Batching

Note: This document is being updated to reflect *NeuronCore-v2 Architecture* and training support.

Batching refers to the process of grouping multiple inference requests together and processing them as a group. Batching is typically used as an optimization for throughput at the expense of higher latency. Batching is usually implemented on a layer-by-layer basis, which allows for each set of weights in a given layer to be reused for each inference in the batch before needing to retrieve additional new weights. This enables Neuron to better amortize the cost of reading weights from the external memory (i.e. read weights from the memory once, and use them in multiple calculations), and thus improve the overall hardware efficiency.

The concept of batched inference is illustrated in the example below, with a single NeuronCore performing batched computation of a 3 layer neural network with a batch-size of 4. The NeuronCore reads weights from the external memory, and then performs the corresponding computations for all 4 inference-requests, before reading the next set of weights, thus better amortizing the cost of reading the weights from the memory.

Neuron uses an ahead-of-time compiler, so to enable batching in Neuron, a model should be explicitly compiled for a target batch-size by setting the input-tensor batch dimension accordingly. Users are encouraged to evaluate multiple batch sizes, in order to determine the optimal latency/throughput deployment-point (which is model/application dependent).

During inference, dynamic batching can be used to process a larger client-side inference batch-size, and allow the framework to automatically break up the user-batch into smaller batch sizes, to match the compiled batch-size. This technique increases the achievable throughput by hiding the framework-to-neuron overhead, and amortizing it over a larger batch size. To enable dynamic batching in TensorFlow, user would set the argument `dynamic_batch_size=True` during call to `tfn.saved_model.compile` method.

For example, the TensorFlow code snippet below enables batching, with dynamic-batching and a batch-size of N=4 when compiling a model to Inferentia target:

```
import numpy as np
import tensorflow.neuron as tfn

# To change the batch size, change the first dimension in example_input
batch_sz = 4
example_input = np.zeros([batch_sz, 224, 224, 3], dtype='float16')

# Note: You can add custom compilation flags such as for mixed precision
# here. None are needed for this model.
compiler_args = []

tfn.saved_model.compile("rn50_fp16",
                        "rn50_fp16_compiled/1",
                        model_feed_dict={'input_1:0': example_input },
                        dynamic_batch_size=True,
                        compiler_args=compiler_args)
```

The following TensorFlow code snippet shows that the model can accept inference requests with arbitrary batch size:

```
import tensorflow as tf
import tensorflow.neuron as tfn

predictor = tf.contrib.predictor.from_saved_model("rn50_fp16_compiled/1")
rt_batch_sz_list = [1, 4, 7, 8, 1024]
for rt_batch_sz in rt_batch_sz_list:
    example_input = np.zeros([rt_batch_sz, 224, 224, 3], dtype='float16')
    model_feed_dict = {'input_1:0': example_input}
    result = predictor(model_feed_dict)
```

Note: Depending on the neural network size, Neuron will have a maximum batch size that works optimally on Inferentia. If an unsupported batch size is used, an internal compiler error message will be displayed. A simple way to explore optimal batch size for your specific model is to increment the batch size from 1 upward, one at a time, and test application performance.

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

18.4 NeuronCore Pipeline

The Neuron software feature referred to as a NeuronCore Pipeline refers to the process of sharding a compute-graph across multiple NeuronCores, caching the model parameters in each core's on-chip memory (cache), and then streaming inference requests across the cores in a pipelined manner. Based on the number of NeuronCores selected, the model might get seamlessly sharded across up-to 16 Inferentia devices (i.e. 64 NeuronCores). This enables users to optimize for both throughput and latency, as it enables the NeuronCores to process neural-networks with locally cached data and avoid the cost of accessing external memory.

One benefit to this approach is that NeuronCore Pipeline can typically hit maximal hardware efficiency without the need for batching (e.g. BERT, ResNet50).

For maximal performance, users should choose an instance-size that can cache the entire model by using sufficient NeuronCores. Inf1 instance types have different number of Inferentia devices, each of which has 4 NeuronCores, as shown here https://aws.amazon.com/ec2/instance-types/inf1/

To enable the NeuronCore Pipeline optimization, the compiler should be invoked with the following flags: `--neuroncore-pipeline-cores N`. The number of NeuronCores is typically chosen to be the minimal number that can fit the entire model, which is currently done through a trial-and-error process (compiling to different number of cores and looking for compilation success/failure message). This process will be automated in the future. A simple formula to help define the number of NeuronCores that may be an appropriate choice is

```
neuroncore-pipeline-cores = 4 * round( number-of-weights-in-model/(2 * 10^7) )
```

This allocates a set of NeuronCores based on the size of the given model's weights and normalizes to multiples of 4 so it uses full Inferentias.

The code snippet below shows how to compile a model with NeuronCore Pipeline for 16 NeuronCores (instance size inf1.6xlarge).

```
import numpy as np
import tensorflow.neuron as tfn

example_input = np.zeros([1,224,224,3], dtype='float16')
tfn.saved_model.compile("rn50_fp16",
                        "rn50_fp16_compiled/1",
                        model_feed_dict={'input_1:0' : example_input },
                        compiler_args = ['--neuroncore-pipeline-cores', '16'])
```

This document is relevant for: `Inf1`

This document is relevant for: `Trn1`

18.5 Neuron Persistent Cache

PyTorch Neuron (`torch-neuronx`) uses `torch-xla`, and `torch-xla` operates in lazy mode. In other words, every operation in training script is recorded in a graph. The graph is executed only when the results are requested by the user when they use `print` or `xm.mark_step`. Requesting results tells `torch-xla` that the recorded graph needs to be executed.

Before executing the graph, `torch-xla` would call Neuron Compiler (`neuronx-cc`) to compile the graph into Neuron specific graph. Then the graph is executed on the *NeuronCore/s*. Compiling the graph involves running optimizations that can make use of the *NeuronCore/s* efficiently. Running these optimizations can be expensive and can result in long compile times. To save the users from compiling these graphs at every iteration, `torch-xla` maintains an in-memory cache called Just in Time (JIT) cache. When the user re-runs the same graph (eg. 2nd iteration of the training run), torch-xla would check in this JIT cache and re-use the cached compilation result, thereby avoiding the wait times.

Since the JIT cache is an in-memory cache, it needs to be constructed every time the training script is run. Hence, if the user re-runs the training script, a new JIT cache is created. This causes a compilation for the first training graph. To avoid such compilations across training runs, PyTorch Neuron (`torch-neuronx`) has built an on-disk `Neuron Persistent Cache`. Since this cache is on-disk, its persistent across training runs. So now, when a graph is compiled for the fist time, the compilation result is saved in `Neuron Persistent Cache`. When the user re-runs the training script, since the JIT cache is not ready, it would send the graph for compilation. PyTorch Neuron (`torch-neuronx`) would then check if the compiled result is present in the `Neuron Persistent Cache`, if yes, it would return with the compiled result. This on-disk cache thereby avoids compilations across training runs. This cache is enabled by default and the default cache directory is `/var/tmp/neuron-compile-cache`.

Look at the diagram below on the end to end flow:

As seen from the diagram, the operations are recorded in a graph in lazy mode and only when a mark_step is hit, the graph is executed. Before execution, the graph passes through two caches to check if we have compiled the graph sometime in the past. If yes, we reuse the compilation result and execute with it. This avoid duplicate compilations. One thing to note, both JIT cache and Neuron Cache are complementary to each other. JIT cache prevents duplicate compilation within a run and Neuron Cache prevents duplicate compilations across training runs. For example, within a training script, we have a training loop that iterates through the dataset. The first iteration would trace a unique graph and the following iteration would trace a graph that is similar to the first one. In this case, the subsequent iterations would hit the JIT cache and reuse the result. However, to save users from compiling for the first iteration graph, `Neuron Persistent Cache` would be used. In this case, the very first time when the script is run, the `Neuron Persistent Cache` would be updated. Going forward when we re-run the training script, compilation results from

`Neuron Persistent Cache` would be used.

To better understand how `Neuron Persistent Cache` works, consider the example below:

```
import torch
import torch_xla
import torch_xla.core.xla_model as xm
device = xm.xla_device()
t1 = torch.randn(3, 3).to(device)
t2 = t1 / 0.5
x = t2.cpu()
```

Running the above example produces the following logs:

```
2022-07-17 17:59:53.000541: INFO ||NCC_WRAPPER||: No candidate found under /var/tmp/
↪neuron-compile-cache/USER_neuroncc-2.0.0.48+e5fcdf753/MODULE_8742523875190354238.
2022-07-17 17:59:53.000541: INFO ||NCC_WRAPPER||: Cache dir for the neff: /var/tmp/
↪neuron-compile-cache/USER_neuroncc-2.0.0.48+e5fcdf753/MODULE_8742523875190354238/
↪MODULE_0_SyncTensorsGraph.6_8742523875190354238_ip-172-31-43-112.ec2.internal-e7aed083-
↪14217-5e404023e468c/d3d4b823-0872-418e-bf85-390dbce2bec3
.
Compiler status PASS
2022-07-17 18:00:01.000117: INFO ||NCC_WRAPPER||: Exiting with a successfully compiled␣
↪graph
```

Re-running the above script would fetch the graph from the neuron cache and you would see logs as follows:

```
2022-07-17 18:05:37.000179: INFO ||NCC_WRAPPER||: Using a cached neff at /var/tmp/neuron-
↪compile-cache/USER_neuroncc-2.0.0.48+e5fcdf753/MODULE_8742523875190354238/MODULE_0_
↪SyncTensorsGraph.6_8742523875190354238_ip-172-31-43-112.ec2.internal-e7aed083-14217-
↪5e404023e468c/d3d4b823-0872-418e-bf85-390dbce2bec3/MODULE_0_SyncTensorsGraph.6_
↪8742523875190354238_ip-172-31-43-112.ec2.internal-e7aed083-14217-5e404023e468c.neff.␣
↪Exiting with a successfully compiled graph
```

As you can see, the next run picks the compiled graph from cache, thereby saving the compilation time. The cache uses hash of the Neuron compiler flags and XLA graph as the key. If the Neuron compiler version or XLA graph changes, you will see recompilation. Examples of changes that would cause XLA graph change include:

- Model type and size
- Batch size
- Optimizer and optimizer hyperparameters
- Location of xm.mark_step()

All compilation results are saved in the cache. To disable the cache, you can pass `--no_cache` option via NEURON_CC_FLAGS:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --no_cache'
```

To change the cache's root directory, pass `--cache_dir=<root dir>` option via NEURON_CC_FLAGS (the actual cache directory will be in `<root dir>/neuron-compile-cache`):

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_dir=
↪<root dir>'
```

Stale cached compiled graphs (NEFFs) are deleted from the cache whenever the size of cache is above default cache size of 100GB . The deletion order is based on least-recently-used first. To change the cache size, pass `--cache_size=SIZE_IN_BYTES`. For example, to change the cache size to 16 MB:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_
↪size=16777216'
```

A cache entry considered stale if the last used time is older than a time-to-live value, currently default to 30 days. If the last used time is earlier than the time-to-live value, then it is not deleted even if cache size exceeds cache size limit. To change cache time-to-live, set the option `--cache_ttl` to the number of days desired:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --cache_ttl=60'
```

You can change the verbose level of the compiler by adding `log_level` to either `WARNING`, `INFO` or `ERROR`. This can be done as follows:

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --log_
↪level=INFO'
```

Note: All compilation results are saved in the cache. In other words even if there is a failed compilation, its result would be saved in the cache. If you want to retry a failed compilation, you can do so by using `--retry_failed_compilation`.

```
os.environ['NEURON_CC_FLAGS'] = os.environ.get('NEURON_CC_FLAGS', '') + ' --retry_failed_
↪compilation'
```

Setting the above flag, would retry all the failed compilations and save fresh results in the cache.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

18.6 Collective Communication

Collective Communications is an integral component of distributed ML training. Multiple training nodes exchange information during ML training via Collective Communication operators such as all-reduce. Neuron provides hardware support for the execution of Collective Communication with the Neuron SDK responsible for the hardware configuration and for the execution orchestration. Neuron provides the following Collective Communication operators:

- all-reduce
- all-gather
- reduce-scatter

Neuron also provides the following peer to peer operators:

- send
- receive

Support for additional Collective Communication operators might be added in future releases. Neuron devices are connected via NeuronLinks within a single instance and EFA links between instances. All NeuronLinks transfer the data directly between Neuron device and between Neuron devices and EFA devices bypassing the host to achieve high bandwidth and low latency.

On a single instance Neuron devices are connected in a 2D torus topology supporting Collective Communication operators in sets of 2, 8 and 32 ranks. Other set sizes might be supported in future releases. A single instance topology can be further extended across multiple instances using EFA NeuronLinks.

Trn1.32xl 2D torus topology

For example an 8x4 topology on a single instance, such as 8 rank tensor parallel and 4 ranks data parallel can be extended across multiple instances creating a large tensor/data parallel training cluster.

Collective Communication support on Neuron requires installation of 3 separate packages:

- `aws-neuronx-runtime-lib` - supports execution on Neuron, not specific to Collective Communication and is always required
- `aws-neuronx-collectives` - supports Collective Communication execution on a single instance and on multiple instances.
- `efa_installer` - low level libraries and drivers to support Collective Communication execution over EFA, required to support Collective Communication on multiple instances.

ML models need to be compiled by the Neuron compiler before they can be executed on Neuron devices. The result of the compilation is a binary object containing computational instruction and data movement instructions.

Any Collective Communication operators encountered during compilation are converted to the place holder instructions to be filled by the runtime/collectives libraries during load and execution. This approach allows Neuron compiler to be unaware of the specific physical topology connecting Neuron devices. Once a compiled mode is placed on Neuron devices the runtime/collectives libraries generate the appropriate data movement instructions based on the placement.

For example, a different set of instructions is generated when the next rank is connected via NeuronLinks or via EFA. Neuron executes Collective Communication operators using dedicated hardware that is not shared with computational resources. That allows Neuron to execute compute and communication in parallel. For example Neuron can all-reduce gradients of one layer while the gradients for another layer are computed. Overlapping compute and communication can result is lower latency and higher performance.

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

18.7 Neuron Control Flow

Note: This feature is supported in *NeuronCore-v2 Architecture*, the NeuronCore that exists in *Trainium*, however it is still not implemented by the Neuron Compiler.

Stay tuned and follow the *Neuron Roadmap*

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

18.8 Neuron Custom C++ Operators

Note: This feature is supported in *NeuronCore-v2 Architecture*, the NeuronCore that exists in *Trainium*, however it is still not implemented by the Neuron SDK.

Stay tuned and follow the *Neuron Roadmap*

This document is relevant for: `Trn1`

This document is relevant for: `Trn1`

18.9 Neuron Dynamic Shapes

Note: This feature is supported in *NeuronCore-v2 Architecture*, the NeuronCore that exists in *Trainium*, however it is still not implemented by the Neuron Compiler.

Stay tuned and follow the *Neuron Roadmap*

This document is relevant for: `Trn1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

NINETEEN

NEURON APPLICATION NOTES

Neuron 2.x

This document is relevant for: `Inf1`, `Trn1`

19.1 Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)

Neuron release 2.3 is the first release of Neuron 2.x, and which enables GA of the new EC2 Trn1 instances. Neuron release 2.3 extends latest release of Neuron 1.x (Neuron 1.19.2) to add support for deep-learning training on the AWS Trainium chips.

Starting with Neuron release 2.3, developers can run deep learning training workloads on Trn1 instances to save training costs by up to 50% over equivalent GPU-based EC2 instances, while getting the highest training performance in AWS cloud for popular NLP models. Neuron 2.x introduces the new capabilities and major architectural updates to support training neural-networks with the Trn1 instances.

In addition, starting with this release, Neuron introduces new packages, renames several packages, and updates Neuron installation and update instructions. This release also ends the support for Neuron Runtime 1.x.

19.1.1 More about the release

What's New	• rn2.3.0_new • neuron-packages-changes • announce-aws-neuron-github-org • announce-neuron-rtd-eol
Get started with Neuron	• *Get Started with PyTorch Neuron* • *Neuron Quick Links*
Tested workloads and known issues	• rn2.3.0_tested • rn2.3.0-known-issues
Frequently Asked Questions (FAQ)	• neuron2-intro-faq • neuron-training-faq
Troubleshooting	• *PyTorch Neuron Troubleshooting on Trn1* • *Neuron Runtime Troubleshooting on Trn1*
Neuron architecture and features	• *Neuron Architecture* • *Neuron Features*
Neuron Components release notes	• *Neuron Components Release Notes*

This document is relevant for: `Inf1`, `Trn1`

Neuron Runtime library

This document is relevant for: `Inf1`

19.2 Introducing Neuron Runtime 2.x (libnrt.so)

Table of contents

- *What are we changing?*
- *Why are we making this change?*
- *How will this change affect the Neuron SDK?*
  - *Neuron Driver*

- *Neuron Runtime*
- *Neuron framework extensions*
- *TensorFlow model server*
- *Neuron tools*

- *How will this change affect me?*
  - *Neuron installation and upgrade*
  - *Migrate your application to Neuron Runtime 2.x (libnrt.so)*
- *Troubleshooting*
  - *Application fails to start*
  - *Application fails to start although I installed latest* `aws-neuron-dkms`
  - *Application unexpected behavior when upgrading to release* Neuron 1.16.0 *or newer*
  - *Application unexpected behavior when downgrading to releases before* Neuron 1.6.0 *(from* Neuron 1.16.0 *or newer)*
  - *Neuron Core is in use*
- *Frequently Asked Questions (FAQ)*
  - *Do I need to recompile my model to run it with Neuron Runtime 2.x (*`libnrt.so`*)?*
  - *Do I need to change my application launch command?*
  - *Can* `libnrt.so` *and* `neuron-rtd` *co-exist in the same environment?*
  - *Are there Neuron framework versions that will not support Neuron Runtime 2.x (*`libnrt.so`*)?*

19.2.1 What are we changing?

Starting with *Neuron 1.16.0* release, *Neuron Runtime 1.x* (`neuron-rtd`) is entering maintenance mode and is replaced by *Neuron Runtime 2.x*, a shared library named (`libnrt.so`). For more information on Runtime 1.x see maintenance_rtd.

Upgrading to `libnrt.so` simplifies Neuron installation and upgrade process, introduces new capabilities for allocating NeuronCores to applications, streamlines container creation, and deprecates tools that are no longer needed.

This document describes the capabilities of *Neuron Runtime 2.x* in detail, provides information needed for successful installation and upgrade, and provides information needed for successful upgrade of Neuron applications using *Neuron Runtime 1.x* (included in releases before *Neuron 1.16.0*) to *Neuron Runtime 2.x* (included in releases *Neuron 1.16.0* or newer).

19.2.2 Why are we making this change?

Before *Neuron 1.16.0*, Neuron Runtime was delivered as a daemon (`neuron-rtd`), and communicated with Neuron framework extensions through a `gRPC` interface. `neuron-rtd` was packaged as an `rpm` or `debian` package (`aws-neuron-runtime`) and required a separate installation step.

Starting with *Neuron 1.16.0*, *Neuron Runtime 2.x* is delivered as a shared library (`libnrt.so`) and is directly linked to Neuron framework extensions. `libnrt.so` is packaged and installed as part of Neuron framework extensions (e.g. TensorFlow Neuron, PyTorch Neuron or MXNet Neuron), and does not require a separate installation step. Installing Neuron Runtime as part of the Neuron framework extensions simplifies installation and improves the user experience. In addition, since `libnrt.so` is directly linked to Neuron framework extensions, it enables faster communication between the Neuron Runtime and Neuron Frameworks by eliminating the `gRPC` interface overhead.

For more information please see *How will this change affect the Neuron SDK?* and *Migrate your application to Neuron Runtime 2.x (libnrt.so)*.

19.2.3 How will this change affect the Neuron SDK?

Neuron Driver

You need to use latest Neuron Driver. For successful installation and upgrade to *Neuron 1.16.0* or newer, you must install or upgrade to Neuron Driver (`aws-neuron-dkms`) *version 2.1.5.0* or newer. Neuron applications using *Neuron 1.16.0* will fail if they do not detect *Neuron Driver version 2.1.5.0* or newer. For installation and upgrade instructions see install-guide-index.

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

To see details of Neuron component versions please see neuron-release-content.

Important:

For successful installation or update to Neuron 1.16.0 and newer from previous releases:

- Stop Neuron Runtime 1.x daemon (`neuron-rtd`) by running: `sudo systemctl stop neuron-rtd`
- Uninstall `neuron-rtd` by running: `sudo apt remove aws-neuron-runtime` or `sudo yum remove aws-neuron-runtime`
- Install or upgrade to latest Neuron Driver (`aws-neuron-dkms`) by following the install-guide-index instructions.
- Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

Neuron Runtime

- Installation Starting from *Neuron 1.16.0*, Neuron releases will no longer include the `aws-neuron-runtime packages`, and the Neuron Runtime will be part of the Neuron framework extension of choice (TensorFlow Neuron, PyTorch Neuron or MXNet Neuron). Installing any Neuron framework package will install the Neuron Runtime library (`libnrt.so`).
  - For installation and upgrade instructions see install-guide-index.
- **Configuring *Neuron Runtime*** Before *Neuron 1.16.0*, configuring *Neuron Runtime 1.x* was performed through configuration files (e.g. /opt/aws/neuron/config/neuron-rtd.config). Starting from *Neuron 1.16.0*, configuring *Neuron Runtime 2.x* can be done through environment variables, see *Neuron Runtime Configuration* for details.
- **Starting and Stopping *Neuron Runtime*** Before introducing `libnrt.so`, `neuron-rtd` ran as a daemon that communicated through a `gRPC` interface. Whenever `neuron-rtd` took ownership of a Neuron device, it continued owning that device until it was stopped. This created the need to stop `neuron-rtd` in certain cases. With the introduction of `libnrt.so`, stopping and starting the *Neuron Runtime* is no longer needed as it runs inside the context of the application. With *Neuron Runtime 2.x*, the act of starting and stopping a Neuron application will cause `libnrt.so` to automatically claim or release the ownership of the required Neuron devices.
- **NeuronCore Groups (NCG) deprecation** Before the introduction of *Neuron Runtime 2.x*, NeuronCore Group (NCG) has been used to define an execution group of one or more NeuronCores where models can be loaded and executed. It also provided separation between processes.

  With the introduction of *Neuron Runtime 2.x*, the strict separation of NeuronCores into groups is no longer needed and NeuronCore Groups (NCG) is deprecated. see eol-ncg for more information.
- **Running multiple *Neuron Runtimes*** Before the introduction of `libnrt.so`, you needed to run multiple `neuron-rtd` daemons to allocate Neuron devices for each `neuron-rtd` using configuration files. After the introduction of `libnrt.so`, you will no longer need to run multiple `neuron-rtd` daemons to allocate Neuron devices to specific Neuron application . With `libnrt.so` allocation of NeuronCores (Neuron device include multiple NeuronCores) to a particular application is done by using `NEURON_RT_VISIBLE_CORES` or `NEURON_RT_NUM_CORES` environment variables, for example:

  ```
  NEURON_RT_VISIBLE_CORES=0-3 myapp1.py
  NEURON_RT_VISIBLE_CORES=4-11 myapp2.py
  ```

  Or

  ```
  NEURON_RT_NUM_CORES=3 myapp1.py &
  NEURON_RT_NUM_CORES=4 myapp2.py &
  ```

  See *Neuron Runtime Configuration* for details.
- **Logging** Similar to *Neuron Runtime 1.x*, *Neuron Runtime 2.x* logs to syslog (verbose logging). To make debugging easier, *Neuron Runtime 2.x* also logs to the console (error-only logging). Refer to *Neuron Runtime Configuration* to see how to increase or decrease logging verbosity.
- **Multi-process access to NeuronCores** With the introduction of `libnrt.so`, it's no longer possible to load models on the same NeuronCore from multiple processes. Access to the same NeuronCore should be done from the same process. Instead you can load models on the same NeuronCore using multiple threads from the same process.

  Note: For optimal performance of multi-model execution, each NeuronCore should execute single model.

- **Neuron Runtime architecture** *Neuron Runtime 2.x* is delivered as a shared library (`libnrt.so`) and is directly linked to Neuron framework extensions. `libnrt.so` is packaged and installed as part of Neuron framework extensions (e.g. TensorFlow Neuron, PyTorch Neuron or MXNet Neuron), and does not require a separate installation step. Installing Neuron Runtime as part of the Neuron framework extensions simplifies installation and improves the user experience. In addition, since `libnrt.so` is directly linked to Neuron framework extensions, it enables faster communication between the Neuron Runtime and Neuron Frameworks by eliminating the `gRPC` interface overhead.

Neuron framework extensions

Starting from *Neuron 1.16.0*, Neuron framework extensions (TensorFlow Neuron, PyTorch Neuron or MXNet Neuron) will be packaged together with `libnrt.so`. It is required to install the `aws-neuron-dkms` Driver version 2.1.5.0 or newer for proper operation. The `neuron-rtd` daemon that was installed in previous releases no longer works starting with Neuron 1.16.0.

To see details of Neuron component versions please see neuron-release-content.

TensorFlow model server

Starting from *Neuron 1.16.0*, TensorFlow Neuron model server will be packaged together with `libnrt.so` and will expect `aws-neuron-dkms` *version 2.1.5.0* or newer for proper operation.

Note: The TensorFlow Neuron model server included in *Neuron 1.16.0* should run from the directory in which it was installed, as it will not run properly if copied to a different location due to its dependency on `libnrt.so`.

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

Neuron tools

- `neuron-cli` - Starting from *Neuron 1.16.0*, `neuron-cli` enters maintenance mode, see maintenance_neuron-cli for more information.
- `neuron-top` - Starting from *Neuron 1.16.0*, `neuron-top` has a new user interface, see *Neuron Top User Guide* for more information.
- `neuron-monitor` - `neuron-monitor` was updated to support Neuron Runtime 2.x (`libnrt.so`)
  - See *Neuron Monitor User Guide* for a updated user guide of `neuron-monitor`.
  - See neuron-monitor-upg for a list of changes between *Neuron Monitor 2.x* and *Neuron Monitor 1.0*
  - See neuron-monitor-bwc for how you can use *Neuron Monitor 2.x* with *Neuron Runtime 1.x* (`neuron-rtd`) .

19.2.4 How will this change affect me?

Neuron installation and upgrade

As explained in "*How will this change affect the Neuron SDK?*", starting from *Neuron 1.16.0*, `libnrt.so` requires the latest Neuron Driver (`aws-neuron-dkms`), in addition there is no longer the need to install `aws-neuron-runtime`. To install Neuron or upgrade to latest Neuron version, please follow the installation and upgrade instructions below:

- **PyTorch Neuron**
  - *Install PyTorch Neuron (torch-neuron)*.
  - *Update to latest PyTorch Neuron (torch-neuron)*.
- **TensorFlow Neuron**
  - *Install TensorFlow Neuron*.
  - *Update to latest TensorFlow Neuron*.
- **MXNet Neuron**
  - *Install MXNet Neuron*.
  - *Update to latest MXNet Neuron*.

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

Migrate your application to Neuron Runtime 2.x (libnrt.so)

For a successful migration of your application to *Neuron 1.16.0* or newer from previous releases, please make sure you perform the following:

1. **Prerequisite** Please read "*How will this change affect the Neuron SDK?*" section.
2. **Make sure you are not using *Neuron Runtime 1.x* (`aws-neuron-runtime`)**
   - Remove any code that install `aws-neuron-runtime` from any CI/CD scripts.
   - Stop `neuron-rtd` by running: `sudo systemctl stop neuron-rtd`
   - Uninstall `neuron-rtd` by running: `sudo apt remove aws-neuron-runtime` or `sudo yum remove aws-neuron-runtime`
3. **Upgrade to your Neuron Framework of choice:**
   - *Update to latest PyTorch Neuron (torch-neuron)*.
   - *Update to latest TensorFlow Neuron*.
   - *Update to latest MXNet Neuron*.
4. **If you have a code that start and/or stop `neuron-rtd`** Remove any code that start or stop `neuron-rtd` from any CI/CD scripts.
5. **Application running multiple `neuron-rtd`** If your application runs multiple processes and required running multiple `neuron-rtd` daemons:
   - Remove the code that runs multiple `neuron-rtd` daemons.

- Instead of allocating Neuron devices to `neuron-rtd` through configuration files, use `NEURON_RT_VISIBLE_CORES` or `NEURON_RT_NUM_CORES` environment variables to allocate NeuronCores. See *Neuron Runtime Configuration* for details.

If you application uses `NEURONCORE_GROUP_SIZES`, see next item.

Note: `NEURON_RT_VISIBLE_CORES` and `NEURON_RT_NUM_CORES` environment variables enables you to allocate NeuronCores to an application. Allocating NeuronCores improves application granularity because Neuron device include multiple NeuronCores.

---

6. **Application running multiple processes using `NEURONCORE_GROUP_SIZES`**

   - Please consider using `NEURON_RT_VISIBLE_CORES` or `NEURON_RT_NUM_CORES` environment variables instead of `NEURONCORE_GROUP_SIZES` as it is being deprecated,

   see *Neuron Runtime Configuration* for details.

   - If you using TensorFlow Neuron (`tensorflow-nueron (TF2.x)`) and you are replacing `NEURONCORE_GROUP_SIZES=AxB` which enables auto multicore replication, please see the new api *TensorFlow Neuron (tensorflow-nueron) Auto Multicore Replication (Experimental)* for usage and documentation.
   - Your application behavior will remain the same as before if you do not set `NEURON_RT_VISIBLE_CORES` and do not set `NEURON_RT_NUM_CORES`.
   - If you are considering migrating to `NEURON_RT_VISIBLE_CORES` or `NEURON_RT_NUM_CORES`, please use the following guidelines:
     - `NEURON_RT_VISIBLE_CORES` takes precedence over `NEURON_RT_NUM_CORES`.
     - If you are migrating to `NEURON_RT_VISIBLE_CORES`:
       * For TensorFlow applications or PyTorch applications make sure that `NEURONCORE_GROUP_SIZES` is unset, or that `NEURONCORE_GROUP_SIZES` allocate the same or less number of NeuronCores allocated by `NEURON_RT_VISIBLE_CORES`.
       * For MXNet applications, setting `NEURONCORE_GROUP_SIZES` and `NEURON_RT_VISIBLE_CORES` environment variables at the same time is not supported. Please use `NEURON_RT_VISIBLE_CORES` only.
       * See *Neuron Runtime Configuration* for more details of how to use `NEURON_RT_VISIBLE_CORES`.
     - If you are migrating to `NEURON_RT_NUM_CORES`:
       * Make sure that `NEURONCORE_GROUP_SIZES` is unset.
       * See *Neuron Runtime Configuration* for more details of how to use `NEURON_RT_NUM_CORES`.

7. **Application running multiple processes accessing same NeuronCore** If your application accesses the same NeuronCore from multiple processes, this is no longer possible with `libnrt.so`. Instead, please modify your application to access the same NeuronCore from multiple threads.

Note: For optimal performance of multi-model execution, each NeuronCore should execute a single model.

---

8. **Neuron Tools**

   - If you are using Neuron Monitor, see neuron-monitor-upg for details.

- If you are using `neuron-cli` please remove any call to `neuron-cli`. For more information, see maintenance_neuron-cli.

9. **Containers** If your application is running within a container, and it previously executed `neuron-rtd` within the container, you need to re-build your container so it will not include or install `aws-neuron-runtime`. See neuron-containers and containers-migration-to-runtime2 for details.

19.2.5 Troubleshooting

Application fails to start

Description

Starting from *Neuron 1.16.0* release, Neuron Runtime (`libnrt.so`) requires *Neuron Driver 2.0* or greater (`aws-neuron-dkms`). Neuron Runtime requires Neuron Driver(`aws-neuron-dkms` package) to access Neuron devices.

If `aws-neuron-dkms` is not installed then the application will fail with an error message on console and syslog that look like the following:

```
NRT:nrt_init      Unable to determine Neuron Driver version. Please check aws-neuron-
↪dkms package is installed.
```

If an old `aws-neuron-dkms` is installed then the application will fail with an error message on console and syslog that look like the following:

```
NRT:nrt_init      This runtime requires Neuron Driver version 2.0 or greater. Please␣
↪upgrade aws-neuron-dkms package.
```

Solution

Please follow the installation steps in install-guide-index to install `aws-neuron-dkms`.

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

Application fails to start although I installed latest `aws-neuron-dkms`

Description

Starting from *Neuron 1.16.0* release, Neuron Runtime (`libnrt.so`) require *Neuron Driver 2.0* or greater (`aws-neuron-dkms`). If an old `aws-neuron-dkms` is installed, the application will fail. You may try to install `aws-neuron-dkms` and still face application failure, this may happen because the `aws-neuron-dkms` installation failed as a result of `neuron-rtd` daemon that is still running .

Solution

- Stop `neuron-rtd` by running: `sudo systemctl stop neuron-rtd`
- Uninstall `neuron-rtd` by running: `sudo apt remove aws-neuron-runtime` or `sudo yum remove aws-neuron-runtime`
- Install `aws-neuron-dkms` by following steps in install-guide-index

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

Application unexpected behavior when upgrading to release *Neuron 1.16.0* or newer

Description

When upgrading to release *Neuron 1.16.0* or newer from previous releases, the OS may include two different versions of *Neuron Runtime*: the `libnrt.so` shared library and `neuron-rtd` daemon. This can happen if the user didn't stop `neuron-rtd` daemon or didn't make sure to uninstall the existing Neuron version before upgrade. In this case the user application may behave unexpectedly.

Solution

If the OS includes two different versions of *Neuron Runtime*, `libnrt.so` shared library and `neuron-rtd` daemon:

- Before running applications that use `neuron-rtd`, restart `neuron-rtd` by calling `sudo systemctl restart neuron-rtd`.
- Before running applications linked with `libnrt.so`, stop `neuron-rtd` by calling `sudo systemctl stop neuron-rtd`.

Application unexpected behavior when downgrading to releases before *Neuron 1.6.0* (from *Neuron 1.16.0* or newer)

Description

When upgrading to release *Neuron 1.16.0* or newer from previous releases, and then downgrading back to releases before *Neuron 1.6.0*, the OS may include two different versions of *Neuron Runtime*: the `libnrt.so` shared library and `neuron-rtd` daemon. This can happen if the user didn't make sure to uninstall the existing Neuron version before upgrade or downgrade. In this case the user application may behave unexpectedly.

Solution

If the OS include two different versions of *Neuron Runtime*, `libnrt.so` shared library and `neuron-rtd` daemon:

- Before running applications that use `neuron-rtd`, restart `neuron-rtd` by calling `sudo systemctl restart neuron-rtd`.
- Before running applications linked with `libnrt.so`, stop `neuron-rtd` by calling `sudo systemctl stop neuron-rtd`.

Neuron Core is in use

Description

A Neuron Core can't be shared between two applications. If an application started using a Neuron Core all other applications trying to use the NeuronCore would fail during runtime initialization with the following message in the console and in syslog:

```
ERROR   NRT:nrt_allocate_neuron_cores               NeuronCore(s) not available -␣
↪Requested:nc1-nc1 Available:0
```

Solution

Terminate the the process using NeuronCore and then try launching the application again.

19.2.6 Frequently Asked Questions (FAQ)

Do I need to recompile my model to run it with Neuron Runtime 2.x (`libnrt.so`)?

No.

Do I need to change my application launch command?

No.

Can `libnrt.so` and `neuron-rtd` co-exist in the same environment?

Although we recommend upgrading to the latest Neuron release, we understand that for a transition period you may continue using `neuron-rtd` for old releases. If you are using Neuron Framework (PyTorch,TensorFlow or MXNet) from releases before *Neuron 1.16.0*:

- Install the latest Neuron Driver (`aws-neuron-dkms`)

Important: Starting Neuron version 2.3, `aws-neuron-dkms` package name is changed to `aws-neuronx-dkms`, see *Introducing first release of Neuron 2.x enabling EC2 Trn1 general availability (GA)*

- For development, we recommend using different environments for Neuron Framework (PyTorch,TensorFlow or MXNet) from releases before *Neuron 1.16.0* and for Neuron Framework (PyTorch,TensorFlow or MXNet) from *Neuron 1.16.0* and newer, if that is not possible, please make sure to stop `neuron-rtd` before executing models using Neuron Framework (PyTorch,TensorFlow or MXNet) from *Neuron 1.16.0* and newer.

- For deployment, when you are ready to upgrade, please upgrade to Neuron Framework (PyTorch,TensorFlow or MXNet) from *Neuron 1.16.0* and newer. see *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for more information.

Warning: Executing models using Neuron Framework (PyTorch,TensorFlow or MXNet) from *Neuron 1.16.0* and newer in an environment where `neuron-rtd` is running may cause undefined behavior. Please make sure to stop `neuron-rtd` before executing models using Neuron Framework (PyTorch,TensorFlow or MXNet) from *Neuron 1.16.0* and newer.

Are there Neuron framework versions that will not support Neuron Runtime 2.x (`libnrt.so`)?

All supported PyTorch Neuron and TensorFlow framework extensions in addition to Neuron MXnet 1.8.0 framework extensions support Neuron Runtime 2.x.

Neuron MxNet 1.5.1 does not support Neuron Runtime 2.x (`libnrt.so`) and has now entered maintenance mode. Please see maintenance_mxnet_1_5 for details.

This document is relevant for: `Inf1`

Performance (Inf1)

This document is relevant for: `Inf1`

19.3 Performance Tuning

Important: NeuronCore Groups (NCG) is deprecated, please see eol-ncg and *Migrate your application to Neuron Runtime 2.x (libnrt.so)* for more details.

This guide is intended to provide the reader with an in-depth understanding on how to optimize neural network performance on Inferentia for both throughput and latency. For simplicity, the guide uses TensorFlow and ResNet-50 model as a teaching example to learn how choosing between different compile-time optimizations (e.g. Batching and NeuronCore Pipeline), as well as model-serving optimizations (e.g. multi-threading and dynamic-batching) improves inference performance.

The following guides are considered prerequisites for this tutorial:

- *Running ResNet50 on Inferentia*
- tensorflow-serving-neurocore-group
- *Neuron Batching*
- *NeuronCore Pipeline*

19.3.1 Batching and pipelining (technical background)

Neuron provides developers with various performance optimization features.

Two of the most widely used features are batching and pipelining. Both techniques aim to keep the data close to the compute engines, but achieve this data locality in different ways. In batching it is achieved by loading the data into an on-chip cache and reusing it multiple times for multiple different model-inputs, while in pipelining this is achieved by caching all model parameters into the on-chip cache across multiple NeuronCores and streaming the calculation across them.

As a general rule of thumb, batching is preferred for applications that aim to optimize throughput and cost at the expense of latency, while pipelining is preferred for applications with high-throughput requirement under a strict latency budget.

19.3.2 Compiling for batching optimization

To enable the batching optimization, we first need to compile the model for a target batch-size. This is done by specifying the batch size in the input tensor's batch dimension during compilation. Users are encouraged to evaluate multiple batch sizes in order to determine the optimal latency/throughput deployment-point, which is application dependent.

For example, the code snippet below enables batching on a ResNet50 model, with a batch-size of 5:

```
import numpy as np
import tensorflow.neuron as tfn

# To change the batch size, change the first dimension in example_input
batch_size = 5
example_input = np.zeros([batch_size,224,224,3], dtype='float16')

tfn.saved_model.compile("rn50_fp16",
                        "rn50_fp16_compiled/1",
                        model_feed_dict={'input_1:0': example_input },
                        dynamic_batch_size=True)
```

Note: Depending on the neural network size, Neuron will have a maximum batch size that works optimally on Inferentia. If an unsupported batch size is used, an internal compiler error message will be displayed. A simple way to explore optimal batch size for your specific model is to increment the batch size from 1 upward, one at a time, and test application performance.

19.3.3 Compiling for pipeline optimization

With NeuronCore Pipeline mode, Neuron stores the model parameters onto the Inferentias' local caches, and streams the inference requests across the available NeuronCores, as specified by the `--neuroncore-pipeline-cores` compiler argument. For example, to compile the model to fit pipeline size of four Inferentia devices (16 NeuronCores) avaliable in the inf1.6xlarge instance size:

```
import numpy as np
import tensorflow.neuron as tfn

compiler_args = ['--neuroncore-pipeline-cores', '16']
example_input = np.zeros([1,224,224,3], dtype='float16')
tfn.saved_model.compile("rn50_fp16",
```

(continues on next page)

(continued from previous page)

```
                                "rn50_fp16_compiled/1",
                                model_feed_dict={'input_1:0': example_input },
                                compiler_args=compiler_args)
```

The minimum number of NeuronCores needed to run a compiled model can be found using Neuron Check Model tool. Please see *Neuron Check Model*.

19.3.4 Model-serving inference optimizations

In order to fully realize the maximum throughput of the compiled model (for either batching and pipelining), users need to launch multiple host CPU threads to feed inputs into the Neuron pipeline. The number of threads need to be larger than the specified maximum number of NeuronCores.

Additionally, dynamic batching (framework optimization currently supported only by TensorFlow-Neuron) can be used to process a larger client-side inference batch-size and the framework automatically breaks up the user-batch into smaller batch sizes to match the compiled batch-size. This technique increases the achievable throughput by hiding the framework-to-neuron overhead, and amortizing it over a larger batch size. To use dynamic batching, set the argument `--dynamic_batch_size=True` during compilation and send larger inference batch size (user inference batch size) that is equal to a multiple of the compiled batch size.

Both of methods can be applied together if that shows improvement in performance. However, multi-threading is always needed as a first step to achieve high throughput. You may need to experiment in order to find the right optimization settings for your application.

By default, the framework sets the number of outstanding inference requests to the total number of NeuronCores plus three. This can be changed by setting the NEURON_MAX_NUM_INFERS environment variable. For example, if the compiled model includes some CPU partitions (as when Neuron compiler decided some operations are more efficient to execute on CPU), the number of threads should be increased to account for the additional compute performed on the CPU. Note that the available instance host memory size should be taken into consideration to avoid out-of-memory errors. As above, you need to experiment in order to find the right optimization settings for your application.

Note: By default the framework allocates NeuronCore Group size to match the size of the compiled model. The size of the model is the number of NeuronCores limit passed to compiler during compilation (`--neuroncore-pipeline-cores` option). For more information see tensorflow-serving-neurocore-group.

19.3.5 Other considerations

Mixed Precision

You can find more defails about performance and accuracy trade offs in *Mixed precision and performance-accuracy tuning (neuron-cc)*.

Operator support

The Neuron Compiler maintains an evolving list of supported operators for each framework: *Neuron Supported operators*

AWS Neuron handles unsupported operators by partitioning the graph into subgraph, and executing them on different targets (e.g. NeuronCore partition, CPU partition). If the entire model can run on Inferentia (i.e. all operators are supported), then the model will be compiled into a single subgraph which will be executed by a NeuronCore Group.

Debug

You can examine the post-compiled model to view the compilation results using the Neuron plugin for TensorBoard. See *Visualize graphs executed on Neuron*.

19.3.6 ResNet-50 optimization example

For an example demonstrating the concepts described here, see *Tensorflow ResNet 50 Optimization Tutorial*

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`

19.4 Parallel Execution using NEURON_RT_NUM_CORES

Important: `NEURONCORE_GROUP_SIZES` will no longer be supported starting Neuron 1.19.0 release if your application is using `NEURONCORE_GROUP_SIZES` please see *Migrate your application to Neuron Runtime 2.x (libnrt.so)* and eol-ncgs-env_2 for more details.

19.4.1 Introduction

Inf1 instances are available with a different number of Inferentia chips, each Inferentia chip is combined of 4 NeuronCores and an Inf1 instance includes 4 to 64 NeuronCores depending on the instance size. This guide will show you how to load one or more compiled models into different consecutive groups of NeuronCores using your framework of choice.

19.4.2 Data Parallel Execution

In PyTorch and TensorFlow, the same compiled model can run in parallel on an Inf1 instance by loading it multiple times, up to the total number of NeuronCores specified in NEURON_RT_NUM_CORES or NEURON_RT_VISIBLE_CORES. For more information about NEURON_RT_NUM_CORES and NEURON_RT_VISIBLE_CORES, please refer to *Neuron Runtime Configuration*.

Running multiple models using single process

To run multiple models using single process, you set the environment variable `NEURON_RT_NUM_CORES` with a comma separated list of number of cores in each group.

You can set `NEURON_RT_NUM_CORES` environment variable at runtime:

```
#!/bin/bash
NEURON_RT_NUM_CORES=13 python your_neuron_application.py
```

Or from within your python process running your models (NOTE: you can only set it once in the same process at the beginning of the script):

```
#!/usr/bin/env python
import os

# Set Environment
os.environ['NEURON_RT_NUM_CORES']='13'

# Load models and run inferences ...
```

The below examples allow you to load 4 models into 4 groups of NeuronCores within one process. For example, if there are 4 models A, B, C, D compiled to 2, 4, 3, and 4 NeuronCores respectively, you directly load the models A, B, C, D in sequence within your TensorFlow or PyTorch Neuron process. This example requires an inf1.6xlarge instance with 16 NeuronCores, as the total number of NeuronCores within the NeuronCore Groups is 13.

In MXNet, the mapping from models to NeuronCores is controlled by context `mx.neuron(neuron_core_index)` where `neuron_core_index` is the NeuronCore index at the start of the group. In the example above, you map model A to `mx.neuron(0)` context, model B to `mx.neuron(2)` context, model C to `mx.neuron(6)` context and model D to `mx.neuron(9)` context. For further details, please refer to *Flexible Execution Group (FlexEG) in Neuron-MXNet*.

For PyTorch

For an automated data parallel solution in PyTorch, please see *Data Parallel Inference on Torch Neuron* for more details.

For Tensorflow

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='13'

# Load models (TF2)
model0 = tf.keras.models.load_model(model0_file) # loaded into the first group of NC0-NC1
model1 = tf.keras.models.load_model(model1_file) # loaded into the second group of NC2-
↪NC5
model2 = tf.keras.models.load_model(model1_file) # loaded into the third group of NC6-NC8
model3 = tf.keras.models.load_model(model1_file) # loaded into the fourth group of NC9-
↪NC12

# run inference by simply calling the loaded model
results0 = model0(inputs0)
results1 = model1(inputs1)
results2 = model2(inputs2)
results3 = model3(inputs3)
```

For MXNet 2.x:

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='13'

# Load models (MXNet)
# loaded into the first group of NC0-NC1
sym, args, aux = mx.model.load_checkpoint(mx_model0_file, 0)
model0 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')
# loaded into the second group of NC2-NC5
sym, args, aux = mx.model.load_checkpoint(mx_model1_file, 0)
model1 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')
# loaded into the third group of NC6-NC8
sym, args, aux = mx.model.load_checkpoint(mx_model2_file, 0)
model2 = sym.bind(ctx=mx.neuron(6), args=args, aux_states=aux, grad_req='null')
# loaded into the fourth group of NC9-NC12
sym, args, aux = mx.model.load_checkpoint(mx_model3_file, 0)
model3 = sym.bind(ctx=mx.neuron(9), args=args, aux_states=aux, grad_req='null')

# run inference by simply calling the loaded model
results0 = model0.forward(data=inputs0)
results1 = model1.forward(data=inputs1)
results2 = model2.forward(data=inputs2)
results3 = model3.forward(data=inputs3)
```

You can identify the NeuronCores used by each application using the `neuron-top` command line tool. For more information about the neuron-top user interface, please see *Neuron Top User Guide*.

```
$ neuron-top
```

Running multiple models using multiple processes

You can also run multiple models in parallel processes, when you set NEURON_RT_NUM_CORES per process:

```
$ NEURON_RT_NUM_CORES=2 python your_1st_neuron_application.py
$ NEURON_RT_NUM_CORES=2 python your_2nd_neuron_application.py
```

The first process automatically selects a first set of 2 unused NeuronCores for its new group. The second process automatically selects a new set of 2 unused NeuronCores for its new group.

Running multiple models on the same NeuronCore group

You can load more than one model in a NeuronCore group within one process. The Neuron runtime will handle switching from one model to the next model within the NeuronCore group when the next model is run within the application. In TensorFlow or PyTorch, simply load the additional models after the initial number of models have been loaded, to fill the NeuronCore groups associated with the process.

For PyTorch:

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='2'

# Load models (PT)
model0 = torch.jit.load(model0_file) # loaded into the first group of NC0-NC1
model1 = torch.jit.load(model1_file) # loaded into the first group of NC0-NC1

# run inference by simply calling the loaded model
results0 = model0(inputs0)
results1 = model1(inputs1)
```

For TensorFlow 2.x:

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='2'
```

(continues on next page)

(continued from previous page)

```
# Load models (TF2)
model0 = tf.keras.models.load_model(model0_file) # loaded into the first group of NC0-NC1
model1 = tf.keras.models.load_model(model1_file) # loaded into the first group of NC0-NC1

# run inference by simply calling the loaded model
results0 = model0(inputs0)
results1 = model1(inputs1)
```

In MXNet, use context `mx.neuron(neuron_core_index)` and use the same NeuronCore start index for the additional models.

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='2'

# Load models (MXNet)
# loaded into the first group of NC0-NC1
sym, args, aux = mx.model.load_checkpoint(mx_model0_file, 0)
model0 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')
# loaded into the first group of NC0-NC1
sym, args, aux = mx.model.load_checkpoint(mx_model1_file, 0)
model1 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')

# run inference by simply calling the loaded model
results0 = model0.forward(data=inputs0)
results1 = model1.forward(data=inputs1)
```

The total `NEURON_RT_NUM_CORES` across all processes cannot exceed the number of NeuronCores available on the instance. For example, on an inf1.xlarge with default configurations where the total number of NeuronCores visible to TensorFlow-Neuron is 4, you can launch one process with `NEURON_RT_NUM_CORES=2` (pipelined) and another process with `NEURON_RT_NUM_CORES=2` (data-parallel).

Examples using `NEURON_RT_NUM_CORES` include:

- *PyTorch example*
- *MXNet example*

19.4.3 Auto Model Replication in TensorFlow Neuron (`tensorflow-neuron`) (Experimental)

Please see the below API documentation to see how to perform automatic replication on multiple cores. Note that the automatic replication will only work on models compiled with pipeline size 1: via `--neuroncore-pipeline-cores=1`. If auto replication is not enabled, the model will default to replicate on up to 4 cores.

Python API (TF 2.x only):

TensorFlow Neuron (tensorflow-nueron) Auto Multicore Replication (Experimental)

CLI API (TF 1.x and TF 2.x):

TensorFlow Neuron TF1.x/TF2.x (tensorflow-nueron TF1.x/TF2.x) Auto Multicore Replication CLI (Experimental)

19.4.4 Auto Model Replication (Being Deprecated)

The Auto Model Replication feature in TensorFlow-Neuron enables you to load the model once and the data parallel replication would happen automatically. This reduces framework memory usage as you are not loading the same model multiple times. This feature is experimental and available in TensorFlow-Neuron only.

To enable Auto Model Replication, set NEURONCORE_GROUP_SIZES to Nx1 where N is the desired replication count (the number of NeuronCore groups, each group has size 1). For example, NEURONCORE_GROUP_SIZES=8x1 would automatically replicate the single-NeuronCore model 8 times.

```
os.environ['NEURONCORE_GROUP_SIZES'] = '4x1'
```

or

```
NEURONCORE_GROUP_SIZES=4x1 python3 application.py
```

When NEURONCORE_GROUP_SIZES is not set, the default is 4x1 where a single-NeuronCore model is replicated 4 times on any sized inf1 machine.

This feature is only available for models compiled with neuroncore-pipeline-cores set to 1 (default).

You will still need to use threads in the scaffolding code to feed the loaded replicated model instance in order to achieve high throughput.

Example of auto model replication: *Running OpenPose on Inferentia*

19.4.5 FAQ

Can I mix data parallel and NeuronCore Pipeline?

Yes. You can compile the model using neuroncore-pipeline-cores option. This tells the compiler to set compilation to the specified number of cores for *NeuronCore Pipeline*. The Neuron Compiler will return a NEFF which fits within this limit. See the *Neuron compiler CLI Reference Guide (neuron-cc)* on how to use this option.

For example, on an inf1.2xlarge, you can load two model instances, each compiled with neuroncore-pipeline-cores set to 2, so that they can run in parallel. The model instances can be loaded from different saved models or from the same saved model.

Can I have a mix of multiple models in one Neuroncore group and single model in another one Neuroncore group?

Currently, you can do this in MXNet by setting up two Neuroncore groups, then load for example multiple models in one NCG using context mx.neuron(0), and load single model in the second NCG using context mx.neuron(2). You can also load single model in the first NCG and multiple models in the second NCG. For example:

```
# Set Environment
os.environ['NEURON_RT_NUM_CORES']='6'

# Load models (MXNet)
# loaded into the first group of NC0-NC1
sym, args, aux = mx.model.load_checkpoint(mx_model0_file, 0)
model0 = sym.bind(ctx=mx.neuron(0), args=args, aux_states=aux, grad_req='null')
# loaded into the second group of NC2-NC5
sym, args, aux = mx.model.load_checkpoint(mx_model1_file, 0)
```

(continues on next page)

(continued from previous page)

```
model1 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')
# loaded into the second group of NC2-NC5
sym, args, aux = mx.model.load_checkpoint(mx_model2_file, 0)
model2 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')
# loaded into the second group of NC2-NC5
sym, args, aux = mx.model.load_checkpoint(mx_model3_file, 0)
model3 = sym.bind(ctx=mx.neuron(2), args=args, aux_states=aux, grad_req='null')

# run inference by simply calling the loaded model
results0 = model0.forward(data=inputs0)
results1 = model1.forward(data=inputs1)
results2 = model2.forward(data=inputs2)
results3 = model3.forward(data=inputs3)
```

Loading multiple models in one NCG and single model in another NCG is currently not supported in TensorFlow and PyTorch.

This document is relevant for: `Inf1`

PyTorch Neuron (torch-neuron)

This document is relevant for: `Inf1`

19.5 Running R-CNNs on Inf1

This application note demonstrates how to compile and run Detectron2-based R-CNNs on Inf1. It also provides guidance on how to use profiling to improve the performance of R-CNN models on Inf1.

Table of contents

- *R-CNN Model Overview*
  - *R-CNN Limitations and Considerations on Inferentia (NeuronCore-v1)*
- *Requirements*
- *Installation*
- *Compiling an R-CNN for Inf1*
  - *Create a Detectron2 R-CNN Model*
  - *Profile the Model*
  - *Compiling the ResNet backbone to Inf1*
- *Optimize the R-CNN model*
  - *Compiling the RPN*
  - *Fusing the Backbone and RPN Head*
  - *Compiling the RoI Heads*
- *End-to-end Compilation and Inference*

- *Benchmarking*
- *Other improvements*
  - *For latency sensitive applications:*
  - *For throughput sensitive applications:*

19.5.1 R-CNN Model Overview

Region-based CNN (R-CNN) models are commonly used for object detection and image segmentation tasks. A typical R-CNN architecture is composed of the following components:

- **Backbone:** The backbone extracts features from input images. In some models, the backbone is a Feature Pyramid Network (FPN), which uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. The backbone is commonly a ResNet or Vision Transformer based network.
- **Region Proposal Network (RPN):** The RPN predicts region proposals with a wide range of scales and aspect ratios. RPNs are constructed using convolutional layers and anchor boxes that serve as references for multiple scales and aspect ratios.
- **Region of Interest (RoI):** The RoI component is used to resize the extracted features that have varying size to the same size so that they can be consumed by a fully connected layer. RoI Align is typically used instead of RoI Pooling because RoI Align provides better alignment.

The Detectron2 library provides many popular PyTorch R-CNN implementations, including R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN. This application note will focus on the Detectron2 R-CNN models.

R-CNN Limitations and Considerations on Inferentia (NeuronCore-v1)

R-CNN models can have a few limitations and considerations on Inferentia (NeuronCore-v1). See the *Model Architecture Fit Guidelines* for more information. These limitatins are not applicable to NeuronCore-v2.

19.5.2 Requirements

This application note is intended to be run on an `inf1.2xlarge`. In practice, R-CNN models can be run on any Inf1 instance size.

Verify that this Jupyter notebook is running the Python kernel environment that was set up according to the PyTorch Installation Guide. You can select the kernel from the "Kernel -> Change Kernel" option on the top of this Jupyter notebook page.

19.5.3 Installation

This application note requires the following pip packages:

1. `torch==1.11.0`
2. `torch-neuron`
3. `neuron-cc`
4. `opencv-python`
5. `pycocotools`

6. `torchvision==0.12.0`
7. `detectron2==0.6`

The following section builds `torchvision` from source and installs the `Detectron2` package. It also reinstalls the Neuron packages to ensure version compability.

The the `torchvision` `roi_align_kernel.cpp` kernel is modified to use OMP threading for multithreaded inference on CPU. This significantly improves the performance of RoI Align kernels on Inf1: OMP threading leads to a 2 - 3x RoI Align latency reduction compared to the default `roi_align_kernel.cpp` kernel configuration.

```
# Install python3.7-dev for pycocotools (a Detectron2 dependency)
!sudo apt install python3.7-dev -y

# Install Neuron packages
!pip config set global.extra-index-url https://pip.repos.neuron.amazonaws.com
!pip uninstall -y torchvision
!pip install --force-reinstall torch-neuron==1.11.0.* neuron-cc[tensorflow] "protobuf==3.
↪20.1" ninja opencv-python

# Change cuda to 10.2 for Detectron2
!sudo rm /usr/local/cuda
!sudo ln -s /usr/local/cuda-10.2 /usr/local/cuda

# Install Torchvision 0.12.0 from source
!git clone -b release/0.12 https://github.com/pytorch/vision.git

# Update the RoI Align kernel to use OMP multithreading
with open('vision/torchvision/csrc/ops/cpu/roi_align_kernel.cpp', 'r') as file:
    content = file.read()

# Enable OMP Multithreading and set the number of threads to 4
old = "// #pragma omp parallel for num_threads(32)"
new = "#pragma omp parallel for num_threads(4)"
content = content.replace(old, new)

# Re-write the file
with open('vision/torchvision/csrc/ops/cpu/roi_align_kernel.cpp', 'w') as file:
    file.write(content)

# Build Torchvision with OMP threading
!cd vision && CFLAGS="-fopenmp" python setup.py bdist_wheel
%pip install vision/dist/*.whl

# Install Detectron2 release v0.6
!python -m pip install 'git+https://github.com/facebookresearch/detectron2.git@v0.6'
```

19.5.4 Compiling an R-CNN for Inf1

By default, R-CNN models are not compilable on Inf1 because they cannot be traced with `torch.jit.trace`, which is a requisite for inference on Inf1. The following section demonstrates techniques for compiling a Detectron2 R-CNN model for inference on Inf1.

Specifically, this section creates a standard Detectron2 R-CNN model using a ResNet-101 backbone. It demonstrates how to use profiling to identify the most compute intensive parts of the R-CNN that should be compiled for accelerated inference on Inf1. It then explains how to manually extract and compile the ResNet backbone (the dominant compute component) and inject the compiled backbone back into the full model for improved performance.

Create a Detectron2 R-CNN Model

We create a Detectron2 R-CNN model using the `COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml` pretrained weights and config file. We also download a sample image from the COCO dataset and run an example inference.

```
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg

def get_model():

    # Configure the R-CNN model
    CONFIG_FILE = "COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"
    WEIGHTS_FILE = "COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"
    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file(CONFIG_FILE))
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(WEIGHTS_FILE)
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
    cfg.MODEL.DEVICE = 'cpu'  # Send to CPU for Neuron Tracing

    # Create the R-CNN predictor wrapper
    predictor = DefaultPredictor(cfg)
    return predictor
```

```
import os
import urllib.request

# Define a function to get a sample image
def get_image():
    filename = 'input.jpg'
    if not os.path.exists(filename):
        url = "http://images.cocodataset.org/val2017/000000439715.jpg"
        urllib.request.urlretrieve(url, filename)
    return filename
```

```
import time
import cv2

# Create an R-CNN model
predictor = get_model()

# Get a sample image from the COCO dataset
```

(continues on next page)

(continued from previous page)

```
image_filename = get_image()
image = cv2.imread(image_filename)

# Run inference and print inference latency
start = time.time()
outputs = predictor(image)
print(f'Inference time: {(time.time() - start):0.3f} s')
```

Profile the Model

We can use the PyTorch Profiler to identify which operators contribute the most to the model's runtime on CPU. Ideally, we can compile these compute intensive operators onto Inf1 for accelerated inference.

```
import torch.autograd.profiler as profiler

with profiler.profile(record_shapes=True) as prof:
    with profiler.record_function("model_inference"):
        predictor(image)
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=30))
```

We see that convolution operators (`aten::convolution`) contribute the most to the inference time. By compiling these convolution operators to Inf1, we can improve performance of the R-CNN model. We can print the R-CNN model architecture to see which layers contain the `aten::convolution` operators:

```
print(predictor.model)
```

We observe that the ResNet FPN backbone (predictor.model.backbone L17-L162) contains the majority of convolution operators in the model. The RPN (predictor.model.proposal_generator L181-L533) also contains a few convolutions. Based on this, we should try to compile the ResNet backbone and RPN onto Inf1 to maximize performance.

Compiling the ResNet backbone to Inf1

In this section we demonstrate how to compile the ResNet backbone to Inf1 and use it for inference.

We "extract" the backbone by accessing it using `predictor.model.backbone`. We compile the backbone using `strict=False` because the backbone outputs a dictionary. We use a fixed input shape (`800 x 800`) for compilation. This will be the input shape that all inputs will be resized to during inference. This section also defines a basic pre-processing function (mostly derived from the Detectron2 R-CNN DefaultPredictor module L308-L318) that reshapes inputs to `800 x 800`.

We also create a `NeuronRCNN` wrapper that we use to inject the compiled backbone back into the model by dynamically replacing the `predictor.model.backbone` attribute with the compiled model.

```
import torch
import torch_neuron

example = torch.rand([1, 3, 800, 800])

# Use `with torch.no_grad():` to avoid a jit tracing issue in the ResNet backbone
with torch.no_grad():
    neuron_backbone = torch_neuron.trace(predictor.model.backbone, example, strict=False)
```

(continues on next page)

(continued from previous page)

```
backbone_filename = 'backbone.pt'
torch.jit.save(neuron_backbone, backbone_filename)
```

```
from detectron2.modeling.meta_arch.rcnn import GeneralizedRCNN
from torch.jit import ScriptModule

class NeuronRCNN(torch.nn.Module):
    """
    Creates a `NeuronRCNN` wrapper that injects the compiled backbone into
    the R-CNN model. It also stores the `size_divisibility` attribute from
    the original backbone.
    """

    def __init__(self, model: GeneralizedRCNN, neuron_backbone: ScriptModule) -> None:
        super().__init__()

        # Keep track of the backbone variables
        size_divisibility = model.backbone.size_divisibility

        # Load and inject the compiled backbone
        model.backbone = neuron_backbone

        # Set backbone variables
        setattr(model.backbone, 'size_divisibility', size_divisibility)

        self.model = model

    def forward(self, x):
        return self.model(x)
```

```
# Create the R-CNN with the compiled backbone
neuron_rcnn = NeuronRCNN(predictor.model, neuron_backbone)
neuron_rcnn.eval()

# Print the R-CNN architecture to verify the backbone is now the
# `neuron_backbone` (shows up as `RecursiveScriptModule`)
print(neuron_rcnn)
```

```
def preprocess(original_image, predictor):
    """
    A basic preprocessing function that sets the input height=800 and
    input width=800. The function is derived from the preprocessing
    steps in the Detectron2 `DefaultPredictor` module.
    """

    height, width = original_image.shape[:2]
    resize_func = predictor.aug.get_transform(original_image)
    resize_func.new_h = 800 # Override height
    resize_func.new_w = 800 # Override width
    image = resize_func.apply_image(original_image)
    image = torch.as_tensor(image.astype("float32").transpose(2, 0, 1))
```

(continues on next page)

(continued from previous page)

```
    inputs = {"image": image, "height": height, "width": width}
    return inputs
```

```
# Get a resized input using the sample image
inputs = preprocess(image, get_model())

# Run inference and print inference latency
start = time.time()
for _ in range(10):
    outputs = neuron_rcnn([inputs])[0]
print(f'Inference time: {((time.time() - start)/10):0.3f} s')
```

```
with profiler.profile(record_shapes=True) as prof:
    with profiler.record_function("model_inference"):
        neuron_rcnn([inputs])
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=30))
```

By running the backbone on Inf1, the overall runtime is already significantly improved. The count and runtime of `aten::convolution` operators is also decreased. We now see a `neuron::forward_v2` operator which is the compiled backbone.

19.5.5 Optimize the R-CNN model

Compiling the RPN

Looking at the profiling, we see that there are still several `aten::convolution`, `aten::linear`, and `aten::addmm` operators that significantly contribute to the model's overall latency. By inspecting the model architecture and code, we can determine that the majority of these operators are contained in the RPN module (predictor.model.proposal_generator L181-L533).

To improve the model performance, we will extract the RPN Head and compile it on Inf1 to increase the number of operators that are running on Inf1. We only compile the RPN Head because the RPN Anchor Generator contains objects that are not traceable with `torch.jit.trace`.

The RPN Head contains five layers that run inference on multiple resized inputs. In order to compile the RPN Head, we create a list of tensors that contain the input ("`features`") shapes that the RPN Head uses at each layer. These tensor shapes can be determined by printing the input shapes in the RPN Head `forward` function (`predictor.model.proposal_generator.rpn_head.forward`).

We also create a new `NeuronRCNN` wrapper that injects both the compiled backbone and RPN Head into the R-CNN model.

```
import math

input_shape = [1, 3, 800, 800] # Overall input shape at inference time

# Create the list example of RPN inputs using the resizing logic from the RPN Head
features = list()
for i in [0, 1, 2, 3, 4]:
    ratio = 1 / (4 * 2**i)
    x_i_h = math.ceil(input_shape[2] * ratio)
    x_i_w = math.ceil(input_shape[3] * ratio)
```

(continues on next page)

(continued from previous page)

```
    feature = torch.zeros(1, 256, x_i_h, x_i_w)
    features.append(feature)
```

```
# Extract and compile the RPN Head
neuron_rpn_head = torch_neuron.trace(predictor.model.proposal_generator.rpn_head,␣
↪[features])
rpn_head_filename = 'rpn_head.pt'
torch.jit.save(neuron_rpn_head, rpn_head_filename)
```

```
class NeuronRCNN(torch.nn.Module):
    """
    Creates a wrapper that injects the compiled backbone and RPN Head
    into the R-CNN model.
    """

    def __init__(self, model: GeneralizedRCNN, neuron_backbone: ScriptModule, neuron_rpn_
↪head: ScriptModule) -> None:
        super().__init__()

        # Keep track of the backbone variables
        size_divisibility = model.backbone.size_divisibility

        # Inject the compiled backbone
        model.backbone = neuron_backbone

        # Set backbone variables
        setattr(model.backbone, 'size_divisibility', size_divisibility)

        # Inject the compiled RPN Head
        model.proposal_generator.rpn_head = neuron_rpn_head

        self.model = model

    def forward(self, x):
        return self.model(x)
```

```
# Create the R-CNN with the compiled backbone and RPN Head
predictor = get_model()
neuron_rcnn = NeuronRCNN(predictor.model, neuron_backbone, neuron_rpn_head)
neuron_rcnn.eval()

# Print the R-CNN architecture to verify the compiled modules show up
print(neuron_rcnn)
```

```
# Run inference and print inference latency
start = time.time()
for _ in range(10):
    outputs = neuron_rcnn([inputs])[0]
print(f'Inference time: {((time.time() - start)/10):0.3f} s')
```

```
with profiler.profile(record_shapes=True) as prof:
    with profiler.record_function("model_inference"):
        neuron_rcnn([inputs])
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=30))
```

By running the compiled backbone and RPN Head on Inf1, the overall runtime is improved. Once again, the number and runtime of `aten::convolution` operators is also decreased. We now see two `neuron::forward_v2` operators which correspond to the compiled backbone and RPN Head.

Fusing the Backbone and RPN Head

It is typically preferable to compile fewer independent models ("subgraphs") on Inf1. Combining models and compiling them as a single subgraph enables the Neuron compiler to perform additional optimizations and reduces the I/O data transfer between CPU and NeuronCores between each subgraph.

In this section, we "fuse" the ResNet backbone and RPN Head into a single model that we compile on Inf1. We create the `NeuronFusedBackboneRPNHead` wrapper to create a compilable model that contains both the ResNet backbone (predictor.model.backbone L17-L162) and RPN Head (predictor.model.proposal_generator L181-L533). We also output the `features` because it is used downstream by the RoI Heads. We compile this `NeuronFusedBackboneRPNHead` wrapper as `neuron_backbone_rpn`. We then create a separate `BackboneRPN` wrapper that we use to inject the `neuron_backbone_rpn` in place of the original backbone and RPN Head. We also copy the remainder of the RPN `forward` code (predictor.model.proposal_generator.forward L431-L480) to create a "fused" backbone + RPN module. Lastly, we re-write the `NeuronRCNN` wrapper to use the fused backbone + RPN module. The `NeuronRCNN` wrapper also uses the `predictor.model forward` code to re-write the rest of the R-CNN model forward function.

```
class NeuronFusedBackboneRPNHead(torch.nn.Module):
    """
    Wrapper to compile the fused ResNet backbone and RPN Head.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.backbone = model.backbone
        self.rpn_head = model.proposal_generator.rpn_head
        self.in_features = model.proposal_generator.in_features

    def forward(self, x):
        features = self.backbone(x)
        features_ = [features[f] for f in self.in_features]
        return self.rpn_head(features_), features
```

```
# Create the wrapper with the combined backbone and RPN Head
predictor = get_model()
backbone_rpn_wrapper = NeuronFusedBackboneRPNHead(predictor.model)
backbone_rpn_wrapper.eval()

# Compile the wrapper
example = torch.rand([1, 3, 800, 800])

with torch.no_grad():
    neuron_backbone_rpn_head = torch_neuron.trace(
        backbone_rpn_wrapper, example, strict=False)
```

(continues on next page)

(continued from previous page)

```
backbone_rpn_filename = 'backbone_rpn.pt'
torch.jit.save(neuron_backbone_rpn_head, backbone_rpn_filename)
```

```
class BackboneRPN(torch.nn.Module):
    """
    Wrapper that uses the compiled `neuron_backbone_rpn` instead
    of the original backbone and RPN Head. We copy the remainder
    of the RPN `forward` code (`predictor.model.proposal_generator.forward`)
    to create a "fused" backbone + RPN module.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.backbone_rpn_head = NeuronFusedBackboneRPNHead(model)
        self._rpn = model.proposal_generator
        self.in_features = model.proposal_generator.in_features

    def forward(self, images):
        preds, features = self.backbone_rpn_head(images.tensor)
        features_ = [features[f] for f in self.in_features]
        pred_objectness_logits, pred_anchor_deltas = preds
        anchors = self._rpn.anchor_generator(features_)

        # Transpose the Hi*Wi*A dimension to the middle:
        pred_objectness_logits = [
            # (N, A, Hi, Wi) -> (N, Hi, Wi, A) -> (N, Hi*Wi*A)
            score.permute(0, 2, 3, 1).flatten(1)
            for score in pred_objectness_logits
        ]
        pred_anchor_deltas = [
            # (N, A*B, Hi, Wi) -> (N, A, B, Hi, Wi) -> (N, Hi, Wi, A, B) -> (N, Hi*Wi*A,␣
↪B)
            x.view(x.shape[0], -1, self._rpn.anchor_generator.box_dim,
                   x.shape[-2], x.shape[-1])
            .permute(0, 3, 4, 1, 2)
            .flatten(1, -2)
            for x in pred_anchor_deltas
        ]

        proposals = self._rpn.predict_proposals(
            anchors, pred_objectness_logits, pred_anchor_deltas, images.image_sizes
        )
        return proposals, features
```

```
class NeuronRCNN(torch.nn.Module):
    """
    Wrapper that uses the fused backbone + RPN module and re-writes
    the rest of the R-CNN `model` `forward` function.
    """
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()

        # Use the fused Backbone + RPN
        self.backbone_rpn = BackboneRPN(model)

        self.roi_heads = model.roi_heads

        self.preprocess_image = model.preprocess_image
        self._postprocess = model._postprocess

    def forward(self, batched_inputs):
        images = self.preprocess_image(batched_inputs)
        proposals, features = self.backbone_rpn(images)
        results, _ = self.roi_heads(images, features, proposals, None)
        return self._postprocess(results, batched_inputs, images.image_sizes)
```

```
# Create the new NeuronRCNN wrapper with the combined backbone and RPN Head
predictor = get_model()
neuron_rcnn = NeuronRCNN(predictor.model)
neuron_rcnn.eval()

# Inject the Neuron compiled models
neuron_rcnn.backbone_rpn.backbone_rpn_head = neuron_backbone_rpn_head

# Print the R-CNN architecture to verify the compiled modules show up
print(neuron_rcnn)
```

```
# Run inference and print inference latency
start = time.time()
for _ in range(10):
    outputs = neuron_rcnn([inputs])[0]
print(f'Inference time: {((time.time() - start)/10):0.3f} s')
```

```
with profiler.profile(record_shapes=True) as prof:
    with profiler.record_function("model_inference"):
        neuron_rcnn([inputs])
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=30))
```

By running the fused backbone + RPN Head on Inf1, the overall runtime is improved again. We now see a single `neuron::forward_v2` operator with a lower runtime than the previous combined runtime of the two separate `neuron::forward_v2` operators.

Compiling the RoI Heads

In this section, we extract and compile part of RoI Heads module (predictor.model.roi_heads L530-L778). This will run most of the remaining `aten::linear` and `aten::addmm` operators on Inf1. We cannot extract the entire RoI Heads module because it contains unsupported operators. Thus, we create a `NeuronBoxHeadBoxPredictor` wrapper that extracts specific parts of the `roi_heads` for compilation. The example input for compilation is the shape of the input into the `self.roi_heads.box_head.forward` function. We write another wrapper, `ROIHead` that combines the compiled `roi_heads` into the rest of the RoI module. The `_forward_box` and `forward` functions are from the `predictor.model.roi_heads` module. We re-write the `NeuronRCNN` wrapper to use the optimized RoI Heads wrapper as well as the fused backbone + RPN module.

```
class NeuronBoxHeadBoxPredictor(torch.nn.Module):
    """
    Wrapper that extracts the RoI Box Head and Box Predictor
    for compilation.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.roi_heads = model.roi_heads

    def forward(self, box_features):
        box_features = self.roi_heads.box_head(box_features)
        predictions = self.roi_heads.box_predictor(box_features)
        return predictions
```

```
# Create the NeuronBoxHeadBoxPredictor wrapper
predictor = get_model()
box_head_predictor = NeuronBoxHeadBoxPredictor(predictor.model)
box_head_predictor.eval()

# Compile the wrapper
example = torch.rand([1000, 256, 7, 7])
neuron_box_head_predictor = torch_neuron.trace(box_head_predictor, example)

roi_head_filename = 'box_head_predictor.pt'
torch.jit.save(neuron_box_head_predictor, roi_head_filename)
```

```
class ROIHead(torch.nn.Module):
    """
    Wrapper that combines the compiled `roi_heads` into the
    rest of the RoI module. The `_forward_box` and `forward`
    functions are from the `predictor.model.roi_heads` module.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.roi_heads = model.roi_heads
        self.neuron_box_head_predictor = NeuronBoxHeadBoxPredictor(model)

    def _forward_box(self, features, proposals):
        features = [features[f] for f in self.roi_heads.box_in_features]
        box_features = self.roi_heads.box_pooler(
```

(continues on next page)

(continued from previous page)

```
            features, [x.proposal_boxes for x in proposals])
        predictions = self.neuron_box_head_predictor(box_features)
        pred_instances, _ = self.roi_heads.box_predictor.inference(
            predictions, proposals)
        return pred_instances

    def forward(self, images, features, proposals, targets=None):
        pred_instances = self._forward_box(features, proposals)
        pred_instances = self.roi_heads.forward_with_given_boxes(
            features, pred_instances)
        return pred_instances, {}
```

```
class NeuronRCNN(torch.nn.Module):
    """
    Wrapper that uses the fused backbone + RPN module and the optimized RoI
    Heads wrapper
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()

        # Create fused Backbone + RPN
        self.backbone_rpn = BackboneRPN(model)

        # Create Neuron RoI Head
        self.roi_heads = ROIHead(model)

        # Define pre and post-processing functions
        self.preprocess_image = model.preprocess_image
        self._postprocess = model._postprocess

    def forward(self, batched_inputs):
        images = self.preprocess_image(batched_inputs)
        proposals, features = self.backbone_rpn(images)
        results, _ = self.roi_heads(images, features, proposals, None)
        return self._postprocess(results, batched_inputs, images.image_sizes)
```

```
# Initialize an R-CNN on CPU
predictor = get_model()

# Create the Neuron R-CNN on CPU
neuron_rcnn = NeuronRCNN(predictor.model)
neuron_rcnn.eval()

# Inject the Neuron compiled models
neuron_rcnn.backbone_rpn.backbone_rpn_head = neuron_backbone_rpn_head
neuron_rcnn.roi_heads.neuron_box_head_predictor = neuron_box_head_predictor
```

```
# Run inference and print inference latency
start = time.time()
for _ in range(10):
```

(continues on next page)

(continued from previous page)

```
    outputs = neuron_rcnn([inputs])[0]
print(f'CPU Inference time: {((time.time() - start)/10):0.3f} s')
```

```
with profiler.profile(record_shapes=True) as prof:
    with profiler.record_function("model_inference"):
        neuron_rcnn([inputs])
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=30))
```

Although the overall latency didn't change significantly, running more of the model on Inf1 instead of CPU will free up CPU resources when multiple models are running in parallel.

19.5.6 End-to-end Compilation and Inference

In this section we provide standalone code that compiles and runs an optimized Detectron2 R-CNN on Inf1. Most of the code in this section is from the previous sections in this application note and it's consolidated here for easy deployment. This section has the following main componennts:

1. Preprocessing and compilation functions
2. Wrappers that extract the R-CNN ResNet backbone, RPN Head, and RoI Head for compilation on Inf1.
3. A `NeuronRCNN` wrapper that creates an optimized end-to-end Detectron2 R-CNN model for inference on Inf1
4. Benchmarking code that runs parallelized inference for optimized throughput on Inf1

Benchmarking

In the benchmarkinng section, we load multiple optimized RCNN models and run them in parallel to maximize throughput.

We use the experimental NeuronCore placement API, `torch_neuron.experimental.neuron_cores_context()`, to ensure all compiled models in an optimized RCNN model are loaded onto the same NeuronCore. Please note that the functionality and API of `torch_neuron.experimental.neuron_cores_context()` might change in future releases.

We define a simple benchmark function that loads four optimized RCNN models onto four separate NeuronCores, runs multithreaded inference, and calculates the corresponding latency and throughput. We benchmark various numbers of loaded models to show the impact of parallelism.

We observe that throughput increases (at the cost of latency) when more models are run in parallel on Inf1. Increasing the number of worker threads also improves throughput.

Other improvements

There are many additional optimizations that can be applied to RCNN models on Inf1 depending on the application:

For latency sensitive applications:

- Each of the five layers in the RPN head can be parallelized to decrease the overall latency.
- The number of OMP Threads can be increased in the ROI Align kernel. Both of these optimizations will improve latency at the cost of decreasing throughput.

For throughput sensitive applications:

- The input batch size can be increased to improve the NeuronCore utilization.

```
import time
import os
import urllib.request
from typing import Any, Union, Callable

import cv2
import numpy as np
from concurrent.futures import ThreadPoolExecutor

import torch
import torch_neuron

from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.modeling.meta_arch.rcnn import GeneralizedRCNN


# ------------------------------------------------------------------------------
# Helper functions
# ------------------------------------------------------------------------------

def get_model():

    # Configure the R-CNN model
    CONFIG_FILE = "COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"
    WEIGHTS_FILE = "COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"
    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file(CONFIG_FILE))
    cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(WEIGHTS_FILE)
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
    cfg.MODEL.DEVICE = 'cpu'  # Send to CPU for Neuron Tracing

    # Create the R-CNN predictor wrapper
    predictor = DefaultPredictor(cfg)
    return predictor


def get_image():

    # Get a sample image
    filename = 'input.jpg'
```

(continues on next page)

(continued from previous page)

```
    if not os.path.exists(filename):
        url = "http://images.cocodataset.org/val2017/000000439715.jpg"
        urllib.request.urlretrieve(url, filename)
    return filename


def preprocess(original_image, predictor):
    """
    A basic preprocessing function that sets the input height=800 and
    input width=800. The function is derived from the preprocessing
    steps in the Detectron2 `DefaultPredictor` module.
    """

    height, width = original_image.shape[:2]
    resize_func = predictor.aug.get_transform(original_image)
    resize_func.new_h = 800 # Override height
    resize_func.new_w = 800 # Override width
    image = resize_func.apply_image(original_image)
    image = torch.as_tensor(image.astype("float32").transpose(2, 0, 1))
    inputs = {"image": image, "height": height, "width": width}
    return inputs


# ---------------------------------------------------------------------------
# Neuron modules
# ---------------------------------------------------------------------------

class NeuronFusedBackboneRPNHead(torch.nn.Module):
    """
    Wrapper to compile the fused ResNet backbone and RPN Head.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.backbone = model.backbone
        self.rpn_head = model.proposal_generator.rpn_head
        self.in_features = model.proposal_generator.in_features

    def forward(self, x):
        features = self.backbone(x)
        features_ = [features[f] for f in self.in_features]
        return self.rpn_head(features_), features


class BackboneRPN(torch.nn.Module):
    """
    Wrapper that uses the compiled `neuron_backbone_rpn` instead
    of the original backbone and RPN Head. We copy the remainder
    of the RPN `forward` code (`predictor.model.proposal_generator.forward`)
    to create a "fused" backbone + RPN module.
    """
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.backbone_rpn_head = NeuronFusedBackboneRPNHead(model)
        self._rpn = model.proposal_generator
        self.in_features = model.proposal_generator.in_features

    def forward(self, images):
        preds, features = self.backbone_rpn_head(images.tensor)
        features_ = [features[f] for f in self.in_features]
        pred_objectness_logits, pred_anchor_deltas = preds
        anchors = self._rpn.anchor_generator(features_)

        # Transpose the Hi*Wi*A dimension to the middle:
        pred_objectness_logits = [
            # (N, A, Hi, Wi) -> (N, Hi, Wi, A) -> (N, Hi*Wi*A)
            score.permute(0, 2, 3, 1).flatten(1)
            for score in pred_objectness_logits
        ]
        pred_anchor_deltas = [
            # (N, A*B, Hi, Wi) -> (N, A, B, Hi, Wi) -> (N, Hi, Wi, A, B) -> (N, Hi*Wi*A,␣
↪B)
            x.view(x.shape[0], -1, self._rpn.anchor_generator.box_dim,
                   x.shape[-2], x.shape[-1])
            .permute(0, 3, 4, 1, 2)
            .flatten(1, -2)
            for x in pred_anchor_deltas
        ]

        proposals = self._rpn.predict_proposals(
            anchors, pred_objectness_logits, pred_anchor_deltas, images.image_sizes
        )
        return proposals, features


class NeuronBoxHeadBoxPredictor(torch.nn.Module):
    """
    Wrapper that extracts the RoI Box Head and Box Predictor
    for compilation.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.roi_heads = model.roi_heads

    def forward(self, box_features):
        box_features = self.roi_heads.box_head(box_features)
        predictions = self.roi_heads.box_predictor(box_features)
        return predictions


class ROIHead(torch.nn.Module):
    """
```

(continues on next page)

(continued from previous page)

```
    Wrapper that combines the compiled `roi_heads` into the
    rest of the RoI module. The `_forward_box` and `forward`
    functions are from the `predictor.model.roi_heads` module.
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()
        self.roi_heads = model.roi_heads
        self.neuron_box_head_predictor = NeuronBoxHeadBoxPredictor(model)

    def _forward_box(self, features, proposals):
        features = [features[f] for f in self.roi_heads.box_in_features]
        box_features = self.roi_heads.box_pooler(
            features, [x.proposal_boxes for x in proposals])
        predictions = self.neuron_box_head_predictor(box_features)
        pred_instances, _ = self.roi_heads.box_predictor.inference(
            predictions, proposals)
        return pred_instances

    def forward(self, images, features, proposals, targets=None):
        pred_instances = self._forward_box(features, proposals)
        pred_instances = self.roi_heads.forward_with_given_boxes(
            features, pred_instances)
        return pred_instances, {}


class NeuronRCNN(torch.nn.Module):
    """
    Wrapper that uses the fused backbone + RPN module and the optimized RoI
    Heads wrapper
    """

    def __init__(self, model: GeneralizedRCNN) -> None:
        super().__init__()

        # Create fused Backbone + RPN
        self.backbone_rpn = BackboneRPN(model)

        # Create Neuron RoI Head
        self.roi_heads = ROIHead(model)

        # Define pre and post-processing functions
        self.preprocess_image = model.preprocess_image
        self._postprocess = model._postprocess

    def forward(self, batched_inputs):
        images = self.preprocess_image(batched_inputs)
        proposals, features = self.backbone_rpn(images)
        results, _ = self.roi_heads(images, features, proposals, None)
        return self._postprocess(results, batched_inputs, images.image_sizes)
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------------------------------------
# Compilation functions
# ------------------------------------------------------------------------------

def compile(
    model: Union[Callable, torch.nn.Module],
    example_inputs: Any,
    filename: str,
    **kwargs
) -> torch.nn.Module:
    """
    Compiles the model for Inf1 if it doesn't already exist and saves it as the provided␣
↪filename.

    model: A module or function which defines a torch model or computation.
    example_inputs: An example set of inputs which will be passed to the
        `model` during compilation.
    filename: Name of the compiled model
    kwargs: Extra `torch_neuron.trace` kwargs
    """

    if not os.path.exists(filename):
        with torch.no_grad():
            compiled_model = torch_neuron.trace(model, example_inputs, **kwargs)
        torch.jit.save(compiled_model, filename)


# ------------------------------------------------------------------------------
# Benchmarking function
# ------------------------------------------------------------------------------

def benchmark(backbone_rpn_filename, roi_head_filename, inputs,
              n_models=4, batch_size=1, n_threads=4, iterations=200):
    """
    A simple benchmarking function that loads `n_models` optimized
    models onto separate NeuronCores, runs multithreaded inference,
    and calculates the corresponding latency and throughput.
    """

    # Load models
    models = list()
    for i in range(n_models):
        with torch_neuron.experimental.neuron_cores_context(i):
            # Create the RCNN with the fused backbone + RPN Head and compiled RoI Heads
            # Initialize an R-CNN on CPU
            predictor = get_model()

            # Create the Neuron R-CNN on CPU
            neuron_rcnn = NeuronRCNN(predictor.model)
            neuron_rcnn.eval()

            # Inject the Neuron compiled models
```

(continues on next page)

(continued from previous page)

```
            neuron_rcnn.backbone_rpn.backbone_rpn_head = torch.jit.load(backbone_rpn_
↪filename)
            neuron_rcnn.roi_heads.neuron_box_head_predictor = torch.jit.load(roi_head_
↪filename)

            models.append(neuron_rcnn)

    # Warmup
    for _ in range(8):
        for model in models:
            model([inputs])

    latencies = []

    # Thread task
    def task(i):
        start = time.time()
        models[i]([inputs])
        finish = time.time()
        latencies.append((finish - start) * 1000)

    begin = time.time()
    with ThreadPoolExecutor(max_workers=n_threads) as pool:
        for i in range(iterations):
            pool.submit(task, i % n_models)
    end = time.time()

    # Compute metrics
    boundaries = [50, 95, 99]
    names = [f'Latency P{i} (ms)' for i in boundaries]
    percentiles = np.percentile(latencies, boundaries)
    duration = end - begin

    # Display metrics
    results = {
        'Samples': iterations,
        'Batch Size': batch_size,
        'Models': n_models,
        'Threads': n_threads,
        'Duration (s)': end - begin,
        'Throughput (inf/s)': (batch_size * iterations) / duration,
        **dict(zip(names, percentiles)),
    }

    print('-' * 80)
    pad = max(map(len, results))
    for key, value in results.items():
        if isinstance(value, float):
            print(f'{key + ":" :<{pad + 1}} {value:0.3f}')
        else:
            print(f'{key + ":" :<{pad + 1}} {value}')
    print()
```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":

    # Create and compile the combined backbone and RPN Head wrapper
    backbone_rpn_filename = 'backbone_rpn.pt'
    predictor = get_model()
    backbone_rpn_wrapper = NeuronFusedBackboneRPNHead(predictor.model)
    backbone_rpn_wrapper.eval()
    example = torch.rand([1, 3, 800, 800])
    compile(backbone_rpn_wrapper, example, backbone_rpn_filename, strict=False)

    # Create and compile the RoI Head wrapper
    roi_head_filename = 'box_head_predictor.pt'
    predictor = get_model()
    box_head_predictor = NeuronBoxHeadBoxPredictor(predictor.model)
    box_head_predictor.eval()
    example = torch.rand([1000, 256, 7, 7])
    compile(box_head_predictor, example, roi_head_filename)

    # Download a sample image from the COCO dataset and read it
    image_filename = get_image()
    image = cv2.imread(image_filename)
    inputs = preprocess(image, get_model())

    # Benchmark the Neuron R-CNN model for various numbers of loaded models
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=1, n_threads=1)
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=1, n_threads=2)
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=2, n_threads=2)
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=2, n_threads=4)
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=4, n_threads=4)
    benchmark(backbone_rpn_filename, roi_head_filename, inputs, n_models=4, n_threads=8)
```

This document is relevant for: `Inf1`

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWENTY

NEURON FAQ

Table of contents

- *Neuron 2.x FAQ*
- *Training Only FAQ*
- *Inference Only FAQ*
- *Runtime FAQ*
- *Compiler FAQ*
- *Neuron Containers*
- *Support*

20.1 Neuron 2.x FAQ

- neuron2-intro-faq

20.2 Training Only FAQ

- neuron-training-faq

20.3 Inference Only FAQ

- neuron-f1-faq
- trouble-shooting-inf1-faq
- tf1_faq
- tf2_faq
- *NeuronPerf*

20.4 Runtime FAQ

- *Neuron Runtime FAQ*

20.5 Compiler FAQ

- *Neuron Compiler FAQ (neuronx-cc)*
- *Neuron Compiler FAQ (neuron-cc)*

20.6 Neuron Containers

- *Neuron Containers FAQ*

20.7 Support

- neuron_roadmap_faq
- contribute-faq

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWENTYONE

TROUBLESHOOTING GUIDE

Table of contents

- *Training Only Troubleshooting*
- *Inference Only Troubleshooting*
- *Containers Troubleshooting*
- *Setup Troubleshooting*

21.1 Training Only Troubleshooting

- *PyTorch Neuron for Training*
- *Neuron Runtime Troubleshooting on Trn1*

21.2 Inference Only Troubleshooting

- *PyTorch Neuron for Inference*
- *Neuron Runtime Troubleshooting on Inf1*
- *NeuronPerf*
- *MXNet Neuron*

21.3 Containers Troubleshooting

- *Containers*

21.4 Setup Troubleshooting

- neuron-setup-troubleshooting

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWENTYTWO

RELEASE DETAILS

- *What's New*
- Release Content

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWENTYTHREE

ROADMAP

The AWS Neuron feature roadmap provides visibility onto what we are working on in terms of functional and performance in the near future. We hope this will help you better plan how to use Neuron with your products. We'd love to get our customers feedback as well, to help us ensure we are working on the most important requests.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

CHAPTER

TWENTYFOUR

SUPPORT

This document is relevant for: `Inf1`, `Trn1`

24.1 SDK Maintenance Policy

Table of Contents

- *Overview*
- *Neuron SDK*
- *Dependency Software*
- *Neuron SDK version life-cycle*
- *Dependency Software version life-cycle*
- *Communication*

24.1.1 Overview

This document outlines the maintenance policy for AWS Neuron Software Development Kit (SDK) and its underlying dependencies. AWS regularly provides the Neuron SDK with updates that may contain support for new or updated APIs, new features, enhancements, bug fixes, security patches, or documentation updates. Updates may also address changes with dependencies, language runtimes, and operating systems. Neuron SDK releases are available as Conda (up to Neuron 1.13.0) and Pip Packages that can be installed within Amazon Machine Images (AMIs).

We recommend users to stay up-to-date with SDK releases to keep up with the latest features, security updates, and underlying dependencies. Continued use of an unsupported SDK version is not recommended and is done at the user's discretion.

24.1.2 Neuron SDK

AWS Neuron is the SDK for AWS Inferentia, the custom designed machine learning chips enabling high-performance deep learning inference applications on EC2 Inf1 instances. Neuron includes a deep learning compiler, runtime and tools that are natively integrated into TensorFlow, PyTorch and MXNet. With Neuron, you can develop, profile, and deploy high-performance inference applications on top of EC2 Inf1 instances.

The Neuron SDK release versions are in the form of X.Y.Z where X represents the major version and Y represent the minor version. Increasing the major version of an SDK indicates that this SDK underwent significant and substantial changes, and some of those changes may not maintain the same programming model. Increasing the minor version of an SDK indicates that this SDK underwent addition of new features, support of new dependency software versions, end-of-support of certain dependency software, enhacement and/or bugfixes. Applications may need to be updated in order for them to work with the newest SDK version. It is important to update major versions carefully and in accordance with the upgrade guidelines provided by AWS.

24.1.3 Dependency Software

Neuron SDK has underlying dependencies, such as language runtimes, operating systems, or third party libraries and machine learning frameworks. These dependencies are typically tied to the language community or the vendor who owns that particular component. The following terms are used to classify underlying dependencies:

- Operating system (OS): Examples include Amazon Linux AMI, Amazon Linux 2.
- Language runtime: Examples include Python.
- Third party library / framework: Examples include PyTorch, TensorFlow, MXNet and ONNX.

Each community or vendor maintains their own versioning policy and publishes their own end-of-support schedule for their product.

24.1.4 Neuron SDK version life-cycle

The life-cycle for Neuron SDK version consists of 3 phases, which are outlined below.

- **Supported (Phase 1)**

  During this phase, AWS will provide critical bugfixes and security patches. Usually AWS will support each Neuron SDK version for at least 12 months, but AWS reserves the right to stop supporting an SDK version before the 12 months period.

  Note: AWS will address new features or Dependency Software updates by publishing a new version with an increment in the Neuron SDK minor version.

- **End-of-Support Announcement (Phase 2)**

  AWS will announce the End-of-Support phase at least 3 months before a specific Neuron SDK version enters End-of-Support phase. During this period, the SDK will continue to be supported.

- **End-of-Support (Phase 3)**

  When a Neuron SDK version reaches end-of support, it will no longer receive critical bugfixes and security patches. Previously published Neuron SDK versions will continue to be available via Conda (up to Neuron 1.13.0) or Pip packages. Use of an SDK version which has reached end-of-support is done at the user's discretion. We recommend users to upgrade to the latest Neuron SDK version.

24.1.5 Dependency Software version life-cycle

The life-cycle for Dependency Software version consists of 4 phases, but there may not be a Phase 3 (Maintenance) period in some cases. The phases are outlined below.

- **Supported (Phase 1)**

  During this phase, Dependency Software version is supported. AWS will provide regular updates, bug fixes and/or security patches to the Dependency Software version, AWS will address those updates and bug fixes by including them in a new Neuron SDK version with an increment in the Neuron SDK minor version. There is no minimum support period for a Dependency Software version.

- **Maintenance and/or End-of-Support Announcement (Phase 2)**

  AWS will announce the Maintenance phase or the End-of-Support phase of Dependency Software version.

  Since each community or vendor maintains their own versioning policy and publishes their own end-of-support schedule for their product, there is no minimum duration to do the announcement before Dependency Software version enters Maintenance phase or End-of-Support phase and in some cases the announcement can happen at the same time when the Dependency Software version enters Maintenance phase or End-of-Support phase.

  During this period, the Dependency Software version will continue to be supported.

- **Maintenance (Phase 3)**

  During the maintenance phase, AWS limits Dependency Software version to address critical bug fixes and security issues only. There is no minimum Maintenance period.

  This phase is optional and AWS will reserve the right to skip it for specific Dependency Software products.

- **End-of-Support (Phase 4)**

  When a Dependency Software version reaches end-of support, it will no longer receive updates or releases. Previously published releases will continue to be available via Conda (up to Neuron 1.13.0) or Pip packages. Use of an SDK which has reached end-of-support is done at the user's discretion. We recommend users to upgrade to the new major version.

  When a Dependency Software version reaches end-of support, it will no longer receive critical bugfixes and security patches. Previously published Dependency Software versions will continue to be available via Neuron SDK Conda (up to Neuron 1.13.0) or Pip packages.

  Use of a Dependency Software version which has reached end-of-support is done at the user's discretion. We recommend users to upgrade to the latest Neuron SDK version that include the latest Dependency Software versions.

Note: AWS reserves the right to stop support for an underlying dependency without a maintenance phase.

24.1.6 Communication

Maintenance and End-Of-Support announcements are communicated as follows:

- Neuron SDK documentation.

To see the list of available Neuron SDK versions and supported Dependency Software versions see neuron-release-content and *What's New* in latest Neuron version.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

24.2 Security Disclosures

If you think you've found a potential security issue, please do not post it in the Issues. Instead, please follow the instructions here (https://aws.amazon.com/security/vulnerability-reporting/) or email AWS security directly (mailto:aws-security@amazon.com).

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

24.3 Contact Us

For support please checkout the Github issues or Neuron AWS forums for an answer, if none of those resources have an answer to your question please open a ticket.

If you have an urgent need for a feature you can also contact us directly at aws-neuron-support@amazon.com.

This document is relevant for: `Inf1`, `Trn1`

This document is relevant for: `Inf1`, `Trn1`

PYTHON MODULE INDEX

p

placement, 901

INDEX

B

benchmark()
    built-in function, 2135
BF16, **2224**
built-in function
    benchmark(), 2135
    compile(), 2134
    get_reports(), 2136
    model_index.append(), 2137
    model_index.copy(), 2137
    model_index.create(), 2137
    model_index.filter(), 2137
    model_index.load(), 2137
    model_index.move(), 2137
    model_index.save(), 2137
    print_reports(), 2136
    torch.neuron.DataParallel(), 896
    torch.neuron.DataParallel.disable_dynamic_batching(), 897
    torch_neuron.trace(), 891
    torch_neuronx.experimental.profiler.profile(), 80
    torch_neuronx.experimental.profiler.profile.start(), 81
    write_csv(), 2136
    write_json(), 2136

C

CCE, **2224**
cFP8, **2224**
Collective Communication Engine, **2222**
compile()
    built-in function, 2134
CustomOps, **2224**

D

DP, **2224**
DPr, **2224**

F

FLOAT32_TO_FLOAT16 (*torch_neuron.Optimization attribute*), 892
FP16, **2224**
FP32, **2224**

G

get_reports()
    built-in function, 2136
GPSIMD Engine, **2222**

I

Inf1, **2221**
Inferentia, **2221**

M

model_index.append()
    built-in function, 2137
model_index.copy()
    built-in function, 2137
model_index.create()
    built-in function, 2137
model_index.filter()
    built-in function, 2137
model_index.load()
    built-in function, 2137
model_index.move()
    built-in function, 2137
model_index.save()
    built-in function, 2137
module
    placement, 901

N

NC, **2224**
ND, **2224**
Neuron Device, **2221**
neuron-cc
    neuron-cc command line option, 2030, 2033
neuron-cc command line option
    neuron-cc, 2030, 2033
neuron-ls
    neuron-ls command line option, 2072
neuron-ls command line option
    neuron-ls, 2072

neuron-monitor
    neuron-monitor command line option, 2056
neuron-monitor command line option
    neuron-monitor, 2056
NeuronCore, **2222**, **2224**
NeuronCore-v1, **2222**
NeuronCore-v2, **2222**
NeuronDevice, **2224**
NeuronLink, **2222**
NeuronLink-v1, **2222**
NeuronLink-v2, **2222**
neuronx-cc
    neuronx-cc command line option, 2019, 2020, 2022
neuronx-cc command line option
    neuronx-cc, 2019, 2020, 2022

P

placement
    module, 901
PP, **2224**
PPr, **2224**
print_reports()
    built-in function, 2136

R

RNE, **2224**
RT, **2224**

S

Scalar Engine, **2222**
ScalEng, **2224**
SR, **2224**
Sync Engine, **2222**
SyncEng, **2224**

T

TensEng, **2224**
Tensor Engine, **2222**
TF32, **2224**
torch.neuron.DataParallel()
    built-in function, 896
torch.neuron.DataParallel.disable_dynamic_batching()
    built-in function, 897
torch_neuron.experimental.multicore_context()
    (*in module placement*), 901
torch_neuron.experimental.neuron_cores_context()
    (*in module placement*), 902
torch_neuron.experimental.set_multicore() (*in module placement*), 903
torch_neuron.experimental.set_neuron_cores()
    (*in module placement*), 904
torch_neuron.Optimization (*built-in class*), 892
torch_neuron.trace()
    built-in function, 891
torch_neuronx.experimental.profiler.profile()
    built-in function, 80
torch_neuronx.experimental.profiler.profile.start()
    built-in function, 81
TP, **2224**
TPr, **2224**
Trainium, **2221**
Trn1, **2221**

V

VecEng, **2224**
Vector Engine, **2222**

W

write_csv()
    built-in function, 2136
write_json()
    built-in function, 2136