Deep Edge - Performance Evaluation of Deep Learning Models on Edge Devices

Introduction:

Deep learning has helped edge devices get smarter and smarter over the last decade. From Intelligent Personal Assistants (IPAs) like Siri, Alexa, Google Now to various text prediction applications, all use deep neural networks to process information and generate results. The state-of-the-art approach utilizes cloud for running these compute intensive machine learning models for the edge devices. The system on chip integration (SoC) in various edge devices can be utilized to run these machine learning models on the edge devices itself. Thus, helping in minimizing latency due to data transfer from edge devices to cloud server. However, there can be various combinations based on number of devices and machine learning models; choosing the best combination of all the possibilities is always a challenge.

This study provides the performance evaluation (in terms of resource utilization- CPU and Memory, processing time, accuracy of prediction and power consumption) by running Image classification machine learning models on 3 main frameworks on various devices mentioned in below sections. Evaluation helps to identify and choose the best combination for the computations.

Devices used and their specifications:

Device	OS	CPU	MEM	GPU/ Accelerator	Storage	Memo
Atomic Pi	Ubuntu 18.04	Intel Atom x5-Z8350 quad core with 2M Cache	2GB RAM, 16GB eMMC	480MHz GPU	SD slot (up to 32GB)	5V / 2A Password: atomicpi123
Raspberry Pi4 Model B	Raspbian9	Cortex A-72 processor, 64 Bit Quad core	4GB RAM	--	Micro SD slot (upto 32G)	5V / 2A Password: raspberry
Edge TPU coprocessor	--	--	--	TPU ML accelerator	--	
Jetson Nano	Ubuntu 18.04	Quad-core ARM A57 @ 1.43 GHz	4 GB 64-bit LPDDR4 25.6 GB/s	128-core Maxwell	microSD (for devkit)	Power: 5V / 2A Passowrd: jetson
Odroid N2	Android 9.0(Pie)	Amlogic S922X Processor (12nm), Quad-core Cortex-A73(1.8Ghz) and Dual-core Cortex-A53(1.9Ghz)	4GB RAM	Mali-G52 GPU	Micro SD slot (32 GB)	12V / Playstore Credentials: Email Id: odroidn2.deepedge@gmail.com Password:DeepEdge123$%
Odroid C2	Android 6.0 (Marshmallow)	Amlogic ARM Cortex-A53 (ARMv8) 1.5Ghz quad core CPUs	2GB RAM	Mali-450 GPU	Micro SD slot (16 GB)	Playstore Credentials: Email Id : deepedge.ugacs@gmail.com Password:Deepedge123$%

List of Frameworks:

Frameworks	Version
TensorFlow with Keras	1.5,2.2.0(AtomicPi); 1.14,2.3.1(RaspPi);1.15,2.3.1(JetsonNano)
PyTorch	1.4(AtomicPi, RaspPi, JetsonNano)
MxNet	1.1(AtomicPi); 2.0(RaspPi);1.6(JetsonNano)
Tensorflow Lite	2.1.0(Accelerator)

List of Models:

MobileNet V1, MobileNet V2, Densenet 161, Squeezenet V1, AlexNet, ResNet50, Inception v3, AlexNet, ResNet 18, GoogleNet, ShuffleNet, VGG16.

Overview of the combinations experimented:

	TensorFlow with Keras	**PyTorch**	**MxNet**	**TensorFlowLite**
MobileNetV1	AtomicPi,RaspberryPi, JetsonNano		AtomicPi,RaspberryPi, JetsonNano	Accelerator, AtomicPi+Accelerator
MobileNetV2	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	JetsonNano,Raspberry Pi	
DenseNet161	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	
ResNet18	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	
ResNet50	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	
VGG16	AtomicPi,RaspberryPi	AtomicPi,RaspberryPi, JetsonNano	RaspberryPi,JetsonNano	
SqueezeNet	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano	
ShuffleNet	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, JetsonNano		
InceptionV3	AtomicPi,RaspberryPi, JetsonNano		AtomicPi,RaspberryPi, JetsonNano	
GoogleNet		AtomicPi,RaspberryPi, JetsonNano		
AlexNet	AtomicPi,RaspberryPi, JetsonNano	AtomicPi,RaspberryPi, Jetson Nano	AtomicPi,RaspberryPi, JetsonNano	

Approach:

Setting up the devices:

Atomic Pi:

Step 1: Download the Ubuntu Bionic LXDE Stand Alone Image from https://www.digital-loggers.com/downloads/index.html#API_IMAGES

Step 2: Download Etcher for Linux. Select Image downloaded, SD card, and flash the image on the drive by clicking on the Flash button on the wizard. Eject and remove the SD card from the system.

Step 3: Insert the SD card, connect the power cable, screen to Atomic Pi. Create password and login to Atomic Pi.

Step 4: Make sure to have Python 3.6+ by using the command `python3 --version` or Install Python by using `sudo apt-get install python3.6`.

Step 5: Check pip version by using the command `pip3 -version`. Install pip by using `sudo apt install python3-pip` and update the device by using `sudo apt-get update.`

Step 6:

```
sudo apt-get install python3-dev default-libmysqlclient-dev build-essential

sudo python3 -m pip install mysqlclient

sudo apt install sysstat

sudo apt install powerstat
```

Raspberry Pi:

Step 1: Download the Raspberry Pi Imager for Ubuntu from https://www.raspberrypi.org/downloads/

Step 2: Run the Imager and select the SD card. Click on the WRITE button.

Step 3: Insert SD card, connect mouse, keyboard, Ethernet cable to the Raspberry Pi. Connect HDMI cable to the screen.

Connect the power supply.

Step 4: NOOBS installer appears offering to install Raspbian OS. Select Raspbian Full [RECOMMENDED] and install.

Step 5: Set up the password and connect to Wi-Fi network if Ethernet cable is not used. Reboot the device after the wizard update is finished.

Step 6: Make sure to have Python 3.6+ by using the command `python3 --version` or Install Python by using `sudo apt-get install python3.6`.

Step 7: Check pip version by using the command `pip3 -version`. Install pip by using `apt install python3-pip` and update the device by using `sudo apt-get update.`

Step 8:

```
sudo apt-get install python3-dev default-libmysqlclient-dev build-essential

sudo python3 -m pip install mysqlclient

sudo apt install sysstat

sudo apt install powerstat
```

Accelerator:

Step 1: Make sure to have Python 3.6+ by using the command `python3 --version` or Install Python by using `sudo apt-get install python3.6`.

Step 2: Check pip version by using the command `pip3 -version`. Install pip by using `apt install python3-pip` and update the device by using `sudo apt-get update`.

Step 3: Add Debian package repository by entering the below command,

```
echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee
/etc/apt/sources.list.d/coral-edgetpu.list

curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo apt-get update
```

Step 4: EdgeTPU runtime library is to be installed by using the below command,

```
sudo apt-get install libedgetpu1-std
```

The device is ready to install the frameworks and run the model.

Jetson Nano:

Step 1: Download Jetson Nano Developer Kit SD Card Image from https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write

Step 2: Download Etcher for Linux. Select Image downloaded, SD card, and flash the image on the drive by clicking on the Flash button on the wizard. Eject and remove the SD card from the system.

Step 3: Insert the SD card, connect the power cable, screen to Jetson Nano. Create password and login to Jetson Nano.

Step 4:

```
sudo apt-get install python3-dev default-libmysqlclient-dev build-essential

sudo python3 -m pip install mysqlclient

sudo apt install sysstat

sudo apt install powerstat
```

Odroid N2:

Step1: Download the android (Pie) OS image from https://wiki.odroid.com/odroid-n2/os_images/android/pie_64_20191018

Step 2: Download Etcher for Linux. Select Image downloaded, SD card, and flash the image on the drive by clicking on the Flash button on the wizard. Eject and remove the SD card from the system.

Step 3: Insert SD card on to Odroid N2 and boot the device.

Step 4: Install Google Play store on Odroid N2 following the steps from https://codewalkerster.blogspot.com/2019/02/how-to-install-google-play-store-on.html and reboot the device.

Odroid C2:

Step1: Download the android (marshmallow) OS image from https://wiki.odroid.com/odroid-c2/os_images/android/marshmallow_v5.7

Step 2: Download Etcher for Linux. Select Image downloaded, SD card, and flash the image on the drive by clicking on the Flash button on the wizard. Eject and remove the SD card from the system.

Step 3: Insert SD card on to Odroid C2 and boot the device.

Step 4: Install Google Play store on Odroid C2 following the steps from https://codewalkerster.blogspot.com/2016/06/how-to-install-google-play-store-on.html and reboot the device.

Installation of frameworks and models:

Tensorflow:

Atomic Pi:

System Requirements: Python 3.6, pip 19.0, Ubuntu 18.04

Install Tensorflow 1.5 version,

```
Sudo pip3 install --upgrade tensorflow==1.5
```

Install Keras 2.3.1 by using the below command,

```
sudo pip3 install keras==2.2.0
```

Install Kerascv library using below command,

```
Sudo pip3 install tensorflow kerascv
```

To go to DeepEdge library in Atomic Pi:

```
type lsblk in the terminal window

cd  /media/atomicpi/EXTERNALSSD/

cd DeepEdge/scripts
```

MobilenetV1: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_mobilenetv1_tf_keras_imagenet.py.

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_mobilenetv1_tf_keras_imagenet.py

Import statements required for running the MobileNetV1 model:

```
from keras.applications.mobilenet import MobileNet

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
mobilenetv1_keras = MobileNet(weights='imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

MobilenetV2: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_mobilenetv2_tf_keras_imagenet.py.

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_mobilenetv2_tf_keras_imagenet.py

Import statements required for running the MobileNetV2 model:

```
from keras.applications.mobilenet_v2 import MobileNetV2

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
mobilenetv2_keras = MobileNetV2(weights= 'imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

SqueezeNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_squeezenet_tf_keras_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_squeezenet_tf_keras_imagenet.py

Import statements required for running the SqueezeNet model:

```
import tensorflow as tf

from keras_squeezenet import SqueezeNet

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
squeezenet = SqueezeNet(weights='imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

The model is imported from the folder keras_squeezenet which is downloaded from the GitHub repository mentioned in [2]

DenseNet161: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_densenet_tf_keras_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_densenet_tf_keras_imagenet.py

Import statements required for running the DenseNet161 model:

```
import tensorflow as tf

from DenseNet_Keras import Densenet
```

Running the model using the script:

```
denseNet = densenet.DenseNetImageNet161(input_shape=(224,224,3))
```

The model is imported from the folder `DenseNet_Keras` which is downloaded from the GitHub repository mentioned in [3]

ShuffleNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_shufflenet_tf_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_shufflenet_tf_keras_imagenet.py

Import statements required for running the ShuffleNet model:

```
from kerascv.model_provider import get_model as kecv_get_model
```

Running the model using the script:

```
shuffleNet = kecv_get_model("shufflenetv2_wd2", pretrained=True)
```

The json file, `imagenet_class_index.json` in the scripts folder holds the labels for 1000 imagenet classes. The json file is used to give the Top-5 accuracy.

ResNet18: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_resnet18_tf_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet18_tf_keras_imagenet.py

Import statements required for running the ResNet18 model:

```
from kerascv.model_provider import get_model as kecv_get_model
from keras.preprocessing import image
```

Running the model using the script:

```
resNet = kecv_get_model("resnet18", pretrained=True)
```

The json file, `imagenet_class_index.json` in the scripts folder holds the labels for 1000 imagenet classes. The json file is used to give the Top-5 accuracy.

VGG16: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_vgg16_tf_keras_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_vgg16_tf_keras_imagenet.py

Import statements required for running the VGG16 model:

```
from keras.applications.vgg16 import VGG16

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
vgg16_keras = VGG16(weights='imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

ResNet50: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_resnet50_tf_keras_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet50_tf_keras_imagenet.py

Import statements required for running the ResNet50 model:

```
from keras.applications.resnet50 import ResNet50

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
resNet_keras = ResNet50(weights='imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

Inception V3: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_inceptionv3_tf_keras_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_inceptionv3_tf_keras_imagenet.py

Import statements required for running the InceptionV3 model:

```
from keras.applications.inception_v3 import InceptionV3

from keras.applications.imagenet_utils import preprocess_input,decode_predictions
```

Running the model using the script:

```
inceptionv3_keras = InceptionV3(weights= 'imagenet')
```

Decode the predictions using the method `decode_predictions` which give the Top 5 accuracy.

AlexNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_alexnet_tf_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_alexnet_tf_keras_imagenet.py

Import statements required for running the AlexNet model:

```
from kerascv.model_provider import get_model as kecv_get_model
```

Running the model using the script:

```
alexNet = kecv_get_model("alexnet", pretrained=True)
```

The json file, `imagenet_class_index.json` in the scripts folder holds the labels for 1000 imagenet classes. The json file is used to give the Top-5 accuracy.

Raspberry Pi:

System Requirements: Python 3.6, pip 19.0, Ubuntu 18.04

Install Tensorflow 1.14 version,

```
sudo pip3 install --upgrade tensorflow==1.14
```

Installation of Keras

```
sudo pip3 install keras
```

Installation of Kerascv is the same as Atomic Pi.

Branch name: raspi

To go to DeepEdge Library:

```
cd Desktop/tf_pi/DeepEdge/scripts
```

AlexNet

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_alexnet_tf_keras_imagenet.py

DenseNet161

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_densenet_tf_keras_imagenet.py

InceptionV3

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_inceptionv3_tf_keras_imagenet.py

MobileNetV1

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_mobilenetv1_tf_keras_imagenet.py

MobileNetV2

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_mobilenetv2_tf_keras_imagenet.py

ResNet18

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_resnet18_tf_keras_imagenet.py

ResNet50

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_resnet50_tf_keras_imagenet.py

ShuffleNet

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_shufflenet_tf_keras_imagenet.py

SqueezeNet

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_squeezenet_tf_keras_imagenet.py

VGG16

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_vgg16_tf_keras_imagenet.py

Accelerator:

System Requirements: Ubuntu 18.04, ARM64 architecture, Python 3.6,

Install Tensorflow Lite Interpreter on the device using the below command,

```
sudo pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-linux_x86_64.whl
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ master Branch □ scripts □ wf_mobnet_tflite_imgclass_accel.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/master/scripts/wf_mobnet_tflite_imgclass_accel.py

Import statements required for running the MobileNetV1 model:

```
import tflite_runtime.interpreter as tflite

from tflite_runtime.interpreter import Interpreter
```

Running the model using the script:

```
interpreter = Interpreter('./models/mobilenet_v1_1.0_224_quant.tflite',experimental_delegates=[tflite.load_delegate('libedgetpu.so.1')])
```

Tensorflow quantized models are downloaded and saved in the models folder.

JetsonNano:

Step 1: Finish setup of the device.

Step 2: System packages required by Tensorflow,

```
sudo apt-get update

sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran

sudo apt-get install python3-pip

sudo pip3 install -U pip testresources setuptools

sudo pip3 install -U numpy==1.16.1 future==0.17.1 mock==3.0.5 h5py==2.9.0 keras_preprocessing==1.0.5 keras_applications==1.0.8 gast==0.2.2 futures protobuf pybind11
```

Step 3: Install specific version of Tensorflow using the below command,

```
sudo pip3 install --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v$JP_VERSION tensorflow==$TF_VERSION+nv$NV_VERSION
```

To go to the Deep Edge library in Jnano:

After you login , right click and click on Open Terminal

```
cd DeepEdge/scripts
```

Converting TensorFlow models to TensorRT:

Step 1: After Installing Tensorflow, Freeze Keras model and convert to TensorRT model

Save the model as .h5 using the script in [31] and then freeze the model to .pb file.

Step 2: Load TensorRT graph and make predictions.

Step3: To rebuild the tensors on jetson nano, include these two lines in the scripts:

```
config.gpu_options.allow_growth = True

config.gpu_options.per_process_gpu_memory_fraction = 0.4
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_mobilenet_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_mobilenet_tfrt_imagenet.py

Import statements required for running the MobileNetV1 model:

```
import tensorflow as tf

from tensorflow.keras.applications.mobilenet import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_mobilenet.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

MobileNetV2: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_mobilenetv2_tfrt_imagenet.py

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_mobilenetv2_tfrt_imagenet.py

Import statements required for running the MobileNetV2 model:

```
import tensorflow as tf

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_mobilenetv2.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

SqueezeNet: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_squeezenet_tfrt_imagenet.py

Link:https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_squeezenet_tfrt_imagenet.py

Import statements required for running the SqueezeNet model:

```
import tensorflow as tf

from tensorflow.keras.applications.imagenet_utils import
preprocess_input,decode_predictions
```

```
from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_squeezenet.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

DenseNet161: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_densenet_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_densenet_tfrt_imagenet.py

Import statements required for running the DenseNet161 model:

```
import tensorflow as tf

from tensorflow.keras.applications.densenet import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_densenet161.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

InceptionV3: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_inceptionv3_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_inceptionv3_tfrt_imagenet.py

Import statements required for running the InceptionV3 model:

```
import tensorflow as tf

from tensorflow.keras.applications.inception_v3 import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_inceptionv3.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

ResNet18: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_resnet18_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet18_tfrt_imagenet.py

Import statements required for running the ResNet18 model:

```
import tensorflow as tf

from tensorflow.keras.applications.resnet import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_resnet18.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

ResNet50: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_resnet50_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet50_tfrt_imagenet.py

Import statements required for running the ResNet50 model:

```
import tensorflow as tf

from tensorflow.keras.applications.resnet50 import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_resnet50.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

ShuffleNet: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_shufflenet_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_shufflenet_tfrt_imagenet.py

Import statements required for running the ShuffleNet model:

```
import tensorflow as tf

from tensorflow.keras.applications.imagenet_utils import
preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_shufflenet.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

VGG16: Script can be found at GitHub repository DeepEdge □ jnano Branch □ scripts □ wl_vgg16_tfrt_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_vgg16_tfrt_imagenet.py

Import statements required for running the VGG16 model:

```
import tensorflow as tf

from tensorflow.keras.applications.vgg16 import preprocess_input,decode_predictions

from tensorflow.keras.preprocessing import image
```

Running the model using the script:

```
graph_file = './models/trt_vgg16.pb'
```

Tensorflow rt model is downloaded and saved in the models folder.

Note: Remote Connection to the UGA Server is not possible on Jetson devices. Cisco AnyConnect does not support the AARCH architecture. To add results in the database, the device must be accessed from the university campus. Database section is commented from the scripts.

Following scripts have been modified:

runner.py, abstract/abstract_workload.py and workload scripts for each model.

To push the logs to the database, uncomment the database section from the above scripts.

The lines to be uncommented are mentioned below in the Jetson Nano Pytorch section

Atomic Pi + Accelerator:

Step 1: Plug in Atomic Pi to the monitor, keyboard and mouse. Connect Accelerator to Atomic Pi.

Step 2: Install Tensorflow lite on Atomic Pi using below command,

```
sudo pip3 install
https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-linux_x86_64.whl
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ master Branch □ scripts □ wf_mobnet_tflite_imgclass_accel.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/master/scripts/wf_mobnet_tflite_imgclass_accel.py

Import statements required for running the MobileNetV1 model:

```
import tflite_runtime.interpreter as tflite

from tflite_runtime.interpreter import Interpreter
```

Running the model using the script:

```
interpreter =
Interpreter('./models/mobilenet_v1_1.0_224_quant.tflite',experimental_delegates=[tflite.lo
ad_delegate('libedgetpu.so.1')])
```

Tensorflow quantized models are downloaded and saved in the models folder.

PyTorch:

Atomic Pi:

System Requirements: Ubuntu 18.04, Python 3.6, pip 19.0

Below command is used to install PyTorch,

```
sudo pip3 install torch==1.4.0+cpu torchvision==0.5.0+cpu -f
https://download.pytorch.org/whl/torch_stable.html
```

MobileNetV2: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_mobilenet_v2_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_mobilenet_v2_pytorch_imagenet.py

Import statements required for running the MobileNetV2 model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
mobilenet = models.mobilenet_v2(pretrained= True)
```

SqueezeNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_squeezenet1_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_squeezenet1_pytorch_imagenet.py

Import statements required for running the SqueezeNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
squeezenet = models.squeezenet1_0(pretrained= True)
```

ShuffleNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_shufflenet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_shufflenet_pytorch_imagenet.py

Import statements required for running the ShuffleNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
shufflenet = models.shufflenet_v2_x1_0(pretrained= True)
```

ResNet18: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_resnet18_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet18_pytorch_imagenet.py

Import statements required for running the ResNet18 model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
resnet = models.resnet18(pretrained= True)
```

ResNet50: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_resnet50_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet50_pytorch_imagenet.py

Import statements required for running the ResNet50 model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
resnet50 = models.resnet50(pretrained= True)
```

DenseNet161: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_densenet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_densenet_pytorch_imagenet.py

Import statements required for running the DenseNet161 model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
densenet = models.densenet161(pretrained= True)
```

VGG16: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_vgg16_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_vgg16_pytorch_imagenet.py

Import statements required for running the VGG16 model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
vgg16 = models.vgg16(pretrained= True)
```

GoogleNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_googlenet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_googlenet_pytorch_imagenet.py

Import statements required for running the GoogleNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
googlenet = models.googlenet(pretrained= True)
```

AlexNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_alexnet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_alexnet_pytorch_imagenet.py

Import statements required for running the AlexNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
alexnet = models.alexnet(pretrained= True)
```

Raspberry Pi:

Step 1: Update Raspbian OS packages by entering the command,

```
sudo apt-get update

sudo apt-get dist-upgrade
```

Step 2: Install dependencies for PyTorch by using command,

```
sudo apt install libopenblas-dev libblas-dev m4 cmake cython python3-dev python3-yaml
python3-setuptools

sudo apt-get install libavutil-dev libavcodec-dev libavformat-dev libswscale-dev
```

Step 3: Download the wheel file from [1]

Step 4: Install PyTorch as below,

```
sudo pip3 install torch-1.4.0a0+f43194e-cp37-cp37m-linux_armv7l.whl
```

```
sudo pip3 install torchvision
```

Importing the models and running the script is the same as Atomic Pi.

Branch Name: raspi

JetsonNano:

Install Pytorch using the below command,

```
wget https://nvidia.box.com/shared/static/ncgzus5o23uck9i5oth2n8n06k340l6k.whl -O
torch-1.4.0-cp36-cp36m-linux_aarch64.whl
```

```
sudo apt-get install python3-pip libopenblas-base
```

```
sudo pip3 install Cython
```

```
sudo pip3 install numpy torch-1.4.0-cp36-cp36m-linux_aarch64.whl
```

```
sudo pip3 install torchvision
```

MobileNetV2: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_mobilenet_v2_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_mobilenet_v2_pytorch_imagenet.py

Import statements required for running the MobileNetV2 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
mobilenet = models.mobilenet_v2(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

ResNet18: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_resnet18_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet18_pytorch_imagenet.py

Import statements required for running the ResNet18 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
resnet = models.resnet18(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

ResNet50: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_resnet50_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet50_pytorch_imagenet.py

Import statements required for running the ResNet50 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
resnet50 = models.resnet50(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

DenseNet161: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_densenet161_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_densenet161_pytorch_imagenet.py

Import statements required for running the DenseNet161 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
densenet = models.densenet161(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

VGG16: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_vgg16_pytorch_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_vgg16_pytorch_imagenet.py

Import statements required for running the VGG16 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
vgg16 = models.vgg16(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database

#img = Image.open(f) ---- Uncomment when using the database
```

SqueezeNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_squeezenet1_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_squeezenet1_pytorch_imagenet.py

Import statements required for running the SqueezeNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
squeezenet = models.squeezenet1_0(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database

#img = Image.open(f) ---- Uncomment when using the database
```

GoogleNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_googlenet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_googlenet_pytorch_imagenet.py

Import statements required for running the GoogleNet model:

```
from torchvision import models,transforms

import torch
```

Running the model using the script:

```
  googlenet = models.googlenet(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database

#img = Image.open(f) ---- Uncomment when using the database
```

ShuffleNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_shufflenet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_shufflenet_pytorch_imagenet.py

Import statements required for running the ShuffleNet model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
shufflenet = models.shufflenet_v2_x1_0(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

AlexNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_alexnet_pytorch_imagenet.py

Link: https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_alexnet_pytorch_imagenet.py

Import statements required for running the AlexNet model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
alexnet = models.alexnet(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

Inception V3: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_inceptionv3_pytorch_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_inceptionv3_pytorch_imagenet.py

Import statements required for running the Inception V3 model:

```
from torchvision import models,transforms
```

```
import torch
```

Running the model using the script:

```
inception = models.inception_v3(pretrained= True)
```

Note: VPN connection to database is not possible for Jetson Nano to retrieve the images from EdgeTeam database when testing the scripts. So input is given from the local system. Please uncomment the below code when you are in University.

In workload scripts:

```
#f = input_data["data_file"] ---- Uncomment when using the database
```

```
#img = Image.open(f) ---- Uncomment when using the database
```

In abstract_workload.py scripts:

```
#input_data = self._input_generator.next_input() ---- Uncomment when using the database
```

In runner.py:

```
'''
    <----- Uncomment  when using the database ----- >

    db_credentials = config["credentials"]["database"][config["db_provider"]]
    db_connection = Connection(db_credentials["host"], db_credentials["username"],
db_credentials["password"], db_credentials["db"])
    '''
'''
    <----- Uncomment  when using the database ----- >
    # retrieve device id
    test_suite_info["device"] = getIdOrThrow(db_connection, "name", config["device_name"],
"devices")
    '''
'''
       <----- Uncomment  when using the database ----- >
        # look for ids
        test_suite_info["model"] = getIdOrThrow(db_connection, "name",
workload.getModel(), "models")
        test_suite_info["framework"] = getIdOrThrow(db_connection, "name",
workload.getFramework(), "frameworks")
        test_suite_info["application"] = getIdOrThrow(db_connection, "name",
workload.getApplication(), "applications")
    '''
'''
        if len(stats) > 0:
            push_stats(con=db_connection, table="runs", test_suite=test_suite_info,
stats=stats)
        else:
            echo ("Nothing to push")
    '''
```

MxNet:

Atomic Pi:

Mxnet for Linux OS having Python3.6 can be installed using pip by below command**,**

```
pip install mxnet==1.5.1
```

Alexnet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_alexnet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_alexnet_mxnet_imagenet.py

Import statements required for running the AlexNet model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()
```

```
alexNet = vision.alexnet(pretrained= True, ctx=ctx)
```

DenseNet161: Script can be found at GitHub repository DeepEdge □ AtomicPii Branch □ scripts □ wl_densenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_densenet_mxnet_imagenet.py

Import statements required for running the DenseNet161 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

denseNet = vision.densenet161(pretrained=True, ctx=ctx)
```

InceptionV3: Script can be found at GitHub repository DeepEdge □ AtomicPiBranch □ scripts □ wl_inceptionv3_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_inceptionv3_mxnet_imagenet.py

Import statements required for running the InceptionV3 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

inceptionModel = vision.inception_v3(pretrained= True, ctx=ctx)
```

ResNet50: Script can be found at GitHub repository DeepEdge □ AtomicPI Branch □ scripts □ wl_resnet50_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet50_mxnet_imagenet.py

Import statements required for running the ResNet50 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet50 = vision.resnet50_v1(pretrained= True, ctx=ctx)
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_mobilenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_mobilenet_mxnet_imagenet.py

Import statements required for running the MobileNetV1 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

mobileNet = vision.mobilenet1_0(pretrained= True, ctx=ctx)
```

ResNet18: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_resnet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_resnet_mxnet_imagenet.py

Import statements required for running the ResNet18 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet = vision.resnet18_v2(pretrained= True, ctx=ctx)
```

SqueezeNet: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_squeezenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_squeezenet_mxnet_imagenet.py

Import statements required for running the SqueezeNet model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

squeezeNet = vision.squeezenet1_0(pretrained= True, ctx=ctx)
```

VGG16: Script can be found at GitHub repository DeepEdge □ AtomicPi Branch □ scripts □ wl_vgg16_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/atomicpi/scripts/wl_vgg16_mxnet_imagenet.py

Import statements required for running the VGG16 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()
```

```
vgg16_model= vision.vgg16(pretrained= True, ctx=ctx)
```

Raspberry Pi:

Step 1: Update the packages and download all the dependencies

```
sudo apt-get update

sudo apt-get -y install git cmake ninja-build build-essential g++-4.9 c++-4.9 liblapack*

libblas* libopencv* libopenblas* python3-dev python-dev virtualenv
```

Step 2: Pull the GitHub repository for MXNet source code.

```
git clone https://github.com/apache/incubator-mxnet.git --recursive

cd incubator-mxnet
```

Step 3: Build the library `libmxnet.so` by using following commands,

```
mkdir -p build && cd build

cmake \

-DUSE_SSE=OFF \

-DUSE_CUDA=OFF \

-DUSE_OPENCV=ON \

-DUSE_OPENMP=ON \

-DUSE_MKL_IF_AVAILABLE=OFF \

-DUSE_SIGNAL_HANDLER=ON \

-DCMAKE_BUILD_TYPE=Release \

-GNinja ..

ninja -j$(nproc)
```

Step 4: Install MxNet Python Bindings,

```
cd python

pip install --upgrade pip

pip install -e .
```

OR

To build a wheel file which can be used to install using pip, use the below command,

```
ci/docker/runtime_functions.sh build_wheel python/ $(realpath build)
```

Branch name: raspi

AlexNet: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_alexnet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_alexnet_mxnet_imagenet.py

Import statements required for running the AlexNet model:

```
import mxnet

from mxnet  import gluon, nd
from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

alexNet = vision.alexnet(pretrained= True, ctx=ctx)
```

DenseNet161: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_densenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_densenet_mxnet_imagenet.py

Import statements required for running the DenseNet161 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

denseNet = vision.densenet161(pretrained=True, ctx=ctx)
```

InceptionV3: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_inceptionv3_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_inceptionv3_mxnet_imagenet.py

Import statements required for running the InceptionV3 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

inceptionModel = vision.inception_v3(pretrained= True, ctx=ctx)
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_mobilenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_mobilenet_mxnet_imagenet.py

Import statements required for running the MobileNeV1 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

mobileNet = vision.mobilenet1_0(pretrained= True, ctx=ctx)
```

MobileNetV2: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_mobilenet2_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_mobilenet2_mxnet_imagenet.py

Import statements required for running the MobileNeV2 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

mobileNet2 = vision.mobilenet_v2_1_0(pretrained= True, ctx=ctx)
```

ResNet18: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_resnet18_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_resnet18_mxnet_imagenet.py

Import statements required for running the ResNet18 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet = vision.resnet18_v2(pretrained= True, ctx=ctx)
```

ResNet50: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_resnet50_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_resnet50_mxnet_imagenet.py

Import statements required for running the ResNet50 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet50 = vision.resnet50_v1(pretrained= True, ctx=ctx)
```

SqueezeNet: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_squeezenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_squeezenet_mxnet_imagenet.py

Import statements required for running the SqueezeNet model:

```
import mxnet
```

```
from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

squeezeNet = vision.squeezenet1_0(pretrained= True, ctx=ctx)
```

VGG16: Script can be found at GitHub repository DeepEdge □ Raspi Branch □ scripts □ wl_vgg16_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/raspi/scripts/wl_vgg16_mxnet_imagenet.py

Import statements required for running the VGG16 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

vgg16_model = vision.vgg16(pretrained= True, ctx=ctx)
```

JetsonNano:

MxNet can be installed by using the below command,

```
sudo pip3 install mxnet-1.6.0-py3-none-any.whl

$ export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

Note: Remote Connection to the UGA Server is not possible on Jetson devices. Cisco AnyConnect does not support the AARCH architecture. To add results in the database, the device must be accessed from the university campus. Database section is commented from the scripts.

Following scripts have been modified:

`runner.py, abstract/abstract_workload.py` and workload scripts for each model.

To push the logs to the database, uncomment the database section from the above scripts.

The lines to be uncommented are mentioned above in the Jetson Nano Pytorch section

AlexNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_alexnet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_alexnet_mxnet_imagenet.py

Import statements required for running the AlexNet model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()
```

```
alexNet = vision.alexnet(pretrained= True, ctx=ctx)
```

DenseNet161: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_densenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_densenet_mxnet_imagenet.py

Import statements required for running the DenseNet161 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

denseNet = vision.densenet161(pretrained=True, ctx=ctx)
```

InceptionV3: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_inceptionv3_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_inceptionv3_mxnet_imagenet.py

Import statements required for running the InceptionV3 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

inceptionModel = vision.inception_v3(pretrained= True, ctx=ctx)
```

MobileNetV2: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_mobilenet2_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_mobilenet2_mxnet_imagenet.py

Import statements required for running the MobileNetV2 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

mobileNet2 = vision.mobilenet_v2_1_0(pretrained= True, ctx=ctx)
```

MobileNetV1: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_mobilenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_mobilenet_mxnet_imagenet.py

Import statements required for running the MobileNetV1 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

mobileNet = vision.mobilenet1_0(pretrained= True, ctx=ctx)
```

ResNet50: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_resnet50_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet50_mxnet_imagenet.py

Import statements required for running the ResNet50 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet50 = vision.resnet50_v1(pretrained= True, ctx=ctx)
```

ResNet18: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_resnet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_resnet_mxnet_imagenet.py

Import statements required for running the ResNet18 model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

resNet = vision.resnet18_v2(pretrained= True, ctx=ctx)
```

SqueezeNet: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_squeezenet_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_squeezenet_mxnet_imagenet.py

Import statements required for running the SqueezeNet model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()
```

```
squeezeNet = vision.squeezenet1_0(pretrained= True, ctx=ctx)
```

VGG16: Script can be found at GitHub repository DeepEdge □ Jnano Branch □ scripts □ wl_vgg16_mxnet_imagenet.py

Link https://github.com/SrujanaMalisetti/DeepEdge/blob/jnano/scripts/wl_vgg16_mxnet_imagenet.py

Import statements required for running the SqueezeNet model:

```
import mxnet

from mxnet  import gluon, nd

from mxnet.gluon.model_zoo import vision
```

Running the model using the script:

```
ctx = mxnet.cpu()

vgg16_model = vision.vgg16(pretrained= True, ctx=ctx)
```

Database:

Link to cloud database: http://172.22.85.19/phpmyadmin/

Name of the DB: EdgeTeam

Tables:

Applications: This table contains a list of applications and data is retrieved by the config.json when script for the model is run on the device.

Datasets: This table contains 100 colorful images used for testing the script for the models.

Devices: This table contains a list of devices used and data is retrieved by the config.json when script for the model is run on the device.

Models: List of all the models used. Data is retrieved by the config.json when the script for the model is run on the device.

Frameworks: List of the frameworks. Data is retrieved by the config.json when the script for the model is run on the device.

GrayScaleImages: Dataset of 100 black and white images to test the ColorNet model on the devices.

Runs: Results of the test run for each model on the device are stored.

Remote Login on Atomic Pi:

Go to internet and select cisco anyconnect
Server: remote.uga.edu
Group 01
Enter username and password for second password: choose phone to receive call on ur phone

Remote Login on Raspberry Pi:

sudo openconnect -u <username> -b remote.uga.edu
group 01
you will get two prompts for passwords:
(1)password < your password>

(2) password: phone

Remote Login on OdroidC2/N2:

Go to the cisco anyconnect mobile app.
Server: remote.uga.edu
Group 01
Enter username and password for second password: choose phone to receive call on ur phone

Running the models:

For First Run: < One Time Activity to set up the database>

Step 1:

In the data folder of the DeepEdge folder of the device. Edit the below json files:

applications.json : name of the application. ImageClassification in our case.

devices.json: name of the device; Eg: AtomicPi, Raspberry Pi , Odroid C2 etc

frameworks: name of the frameworks; Eg: TensorFlow, PyTorch, mxNet

models : name of the machine learning models; Eg: AlexNet, DenseNet etc.

Step 2:

cd to scripts folder inside the DeepEdge folder.

Step 3:

sudo make init

For every run:

Step 1:

cd to scripts folder inside the DeepEdge folder.

Step 2: Open the config.json file and add the workload files in the workloads array :

vim config.json

and edit the workloads array as follows:

“file” : <name of the workload file>

“batch_size”: <size of the batch>

```
   "workloads": [
{
       "file":"wl_shufflenet_tfrt_imagenet.py",
       "batch_size":1
    }
]
```

Note: Any number of workload files can be added in the workloads array(separated by commas) of the config.json. The ability to process multiple workload files and different batch sizes depends on the processing capability of the device.

Step3: Save changes in the config.json file

sudo make

References:

[1] PyTorch- RaspberryPi:
https://github.com/sungjuGit/PyTorch-and-Vision-for-Raspberry-Pi-4B/blob/master/torch-1.4.0a0%2Bf43194e-cp37-cp37m-linux_armv7l.whl

[2] Odroid C2 – Android OS Image: https://wiki.odroid.com/odroid-c2/os_images/android/marshmallow_v5.7

[3] Using Etcher: https://wiki.odroid.com/troubleshooting/odroid_flashing_tools

[4] Playstore Installation on Odroid C2:
https://codewalkerster.blogspot.com/2016/06/how-to-install-google-play-store-on.html

[5] Pytorch Installation on Atomic Pi: https://pytorch.org/get-started/locally/

[6] PyTorch on Jetson Nano 1.4:
https://forums.developer.nvidia.com/t/pytorch-for-jetson-nano-version-1-4-0-now-available/72048

[7] torch vision library: https://pytorch.org/docs/stable/torchvision/index.html

[8]https://pytorch.org/docs/stable/torchvision/models.html#classification

[9]https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

[10] https://pytorch.org/mobile/android/

[11] https://github.com/pytorch/android-demo-app

[12]MxNet Installation on AtomicPi :
https://mxnet.apache.org/get_started?platform=linux&language=python&processor=cpu&environ=pip&iot=raspberry-pi&

[13]MxNet on Raspberry Pi : Build from source:
https://mxnet.apache.org/get_started?platform=devices&iot=raspberry-pi&

[14]MxNet on Jetson Nano 1.6:
https://forums.developer.nvidia.com/t/i-was-unable-to-compile-and-install-mxnet1-5-with-tensorrt-on-the-jetson-nano-is-there-someone-have-compile-it-please-help-me-thank-you/111303#5426042

[15] https://mxnet.apache.org/api/python/docs/api/mxnet/image/index.html#

[16] https://mxnet.apache.org/api/python/docs/tutorials/packages/gluon/image/pretrained_models.html

[17] Tensorflow on Atomic Pi & Raspberry Pi: https://www.tensorflow.org/install/pip

[18]Set up of Jetson Nano ----2.1.0:
https://docs.nvidia.com/deeplearning/frameworks/install-tf-jetson-platform/index.html

[19]https://github.com/rcmalli/keras-squeezenet

[20]https://github.com/titu1994/DenseNet

[21]https://pypi.org/project/kerascv/

[22]https://pypi.org/project/Keras/

[23]https://keras.io/applications/

[24] Jetson Nano-ResNet18: https://github.com/qubvel/classification_models.git

[25] Raspberry OS Installation Using NOOBS: https://www.raspberrypi.org/documentation/installation/noobs.md

[26] Atomic Pi OS Installation: Image Link Ubuntu Bionic LXDE Stand Alone Image
https://www.digital-loggers.com/downloads/index.html#API_IMAGES

[27]https://youtu.be/zM7HwdaUv0o

[28]Jetson Nano OS Installation : JetPack 4.3:
https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write

[29] Odroid N2 OS Installation: Image Link: (Self Installation Link):
https://wiki.odroid.com/odroid-n2/os_images/android/pie_64_20191018

[30]Playstore Installation: https://codewalkerster.blogspot.com/2019/02/how-to-install-google-play-store-on.html

[31]https://www.dlology.com/blog/how-to-run-keras-model-on-jetson-nano/

[32]https://forums.developer.nvidia.com/t/out-of-memory-error-from-tensorflow-any-workaround-for-this-or-do-i-just-need-a-bigger-boat/75082