PyTorch

Release Notes

Table of Contents

Chapter 1. PyTorch Overview

The NVIDIA Deep Learning SDK accelerates widely-used deep learning frameworks such as PyTorch.

PyTorch is a GPU accelerated tensor computational framework with a Python front end. Functionality can be easily extended with common Python libraries such as NumPy, SciPy and Cython. Automatic differentiation is done with a tape-based system at both a functional and neural network layer level. This functionality brings a high level of flexibility and speed as a deep learning framework and provides accelerated NumPy-like functionality.

PyTorch also includes standard defined neural network layers, deep learning optimizers, data loading utilities, and multi-gpu and multi-node support. Functions are executed immediately instead of enqueued in a static graph, improving ease of use and a sophisticated debugging experience.

See `/workspace/README.md` inside the container for information on customizing your PyTorch image. For more information about PyTorch, including tutorials, documentation, and examples, see:

- PyTorch website
- PyTorch project

This document describes the key features, software enhancements and improvements, any known issues, and how to run this container.

Chapter 2. Pulling A Container

About this task

Before you can pull a container from the NGC container registry, you must have Docker installed. For DGX users, this is explained in Preparing to use NVIDIA Containers Getting Started Guide.

For users other than DGX, follow the NVIDIA® GPU Cloud™ (NGC) container registry installation documentation based on your platform.

You must also have access and be logged into the NGC container registry as explained in the NGC Getting Started Guide.

The deep learning frameworks are stored in the following repository where you can find the NGC Docker containers.

`nvcr.io/nvidia`

The deep learning framework containers are stored in the `nvcr.io/nvidia` repository.

Chapter 3. Running PyTorch

Before you begin

Before you can run an NGC deep learning framework container, your Docker environment must support NVIDIA GPUs. To run a container, issue the appropriate command as explained in the Running A Container chapter in the *NVIDIA Containers And Frameworks User Guide* and specify the registry, repository, and tags.

About this task

On a system with GPU support for NGC containers, the following occurs when running a container:

- The Docker engine loads the image into a container which runs the software.
- You define the runtime resources of the container by including additional flags and settings that are used with the command. These flags and settings are described in Running A Container.
- The GPUs are explicitly defined for the Docker container (defaults to all GPUs, but can be specified using `NVIDIA_VISIBLE_DEVICES` environment variable). Starting in Docker 19.03, follow the steps as outlined below. For more information, refer to the nvidia-docker documentation here.

The method implemented in your system depends on the DGX OS version installed (for DGX systems), the specific NGC Cloud Image provided by a Cloud Service Provider, or the software that you have installed in preparation for running NGC containers on TITAN PCs, Quadro PCs, or vGPUs.

Procedure

1. Issue the command for the applicable release of the container that you want. The following command assumes you want to pull the latest container.
   ```
   docker pull nvcr.io/nvidia/pytorch:21.07-py3
   ```
2. Open a command prompt and paste the pull command. The pulling of the container image begins. Ensure the pull completes successfully before proceeding to the next step.
3. Run the container image. To run the container, choose interactive mode or non-interactive mode.

a). **Interactive mode:**

If you have Docker 19.03 or later, a typical command to launch the container is:

```
docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/
pytorch:<xx.xx>-py3
```

If you have Docker 19.02 or earlier, a typical command to launch the container is:

```
nvidia-docker run -it --rm -v local_dir:container_dir nvcr.io/nvidia/pytorch:<xx.xx>-
py3
```

b). **Non-interactive mode:**

If you have Docker 19.03 or later, a typical command to launch the container is:

```
docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/
pytorch:<xx.xx>-py3 <command>
```

If you have Docker 19.02 or earlier, a typical command to launch the container is:

```
nvidia-docker run -it --rm -v local_dir:container_dir nvcr.io/nvidia/pytorch:<xx.xx>-
py3 <command>
```

> **Note:** If you use multiprocessing for multi-threaded data loaders, the default shared memory segment size that the container runs with may not be enough. Therefore, you should increase the shared memory size by issuing either:
>
> `--ipc=host`
>
> or
>
> `--shm-size=<requested memory size>`
>
> in the command line to
>
> `docker run --gpus all`

You might want to pull in data and model descriptions from locations outside the container for use by PyTorch or save results to locations outside the container. To accomplish this, the easiest method is to mount one or more host directories as Docker® data volumes.

Chapter 4. PyTorch Release 21.07

The NVIDIA container image for PyTorch, release 21.07, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.4.0
- cuBLAS 11.5.2.43
- NVIDIA cuDNN 8.2.2.26
- NVIDIA NCCL 2.10.3
- APEX
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.1
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.2.0.0
- Nsight Systems 2021.2.4.12
- TensorRT 8.0.1.6
- TensorBoard 2.5.0
- DALI 1.3
- MAGMA 2.5.2
- DLProf 1.3.0
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext

- TransformerXL
- Jasper
- BERT
- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2

- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.3.1
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.07 is based on NVIDIA CUDA 11.4.0, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.07 is based on 1.10.0a0+ecc3718

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Starting in 21.06, PyProf will no longer be included in the NVIDIA PyTorch container. To profile models in PyTorch, please use NVIDIA Deep Learning Profiler (DLProf). DLProf can help data scientists, engineers and researchers understand and improve performance of their models with visualization via DLProf Viewer in the web browser, or by analyzing text reports. DL Prof is available on NGC or a Python PIP wheel installation.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.07	20.04	NVIDIA CUDA 11.4.0	1.10.0a0+ecc3718	TensorRT 8.0.1.6
21.06		NVIDIA CUDA 11.3.1	1.9.0a0+c3d40fd	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.9.0a0+2ecb2c7	
21.04				
21.03		NVIDIA CUDA 11.2.1	1.9.0a0+df837d0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based

neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The 21.07 release includes libsystemd and libudev versions that have a known vulnerability that was discovered late in our QA process. See CVE-2021-33910 for details. This will be fixed in the next release.

Chapter 5. PyTorch Release 21.06

The NVIDIA container image for PyTorch, release 21.06, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.3.1
- cuBLAS 11.5.1.109
- NVIDIA cuDNN 8.2.1
- NVIDIA NCCL 2.9.9 (optimized for NVLink™)
- APEX
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.1
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.0
- Nsight Systems 2021.2.1.58
- TensorRT 7.2.3.4
- TensorBoard 1.15.5
- DALI 1.2
- MAGMA 2.5.2
- DLProf 1.2.0
- PyProf r21.06
- Tensor Core optimized examples:
    - ResNeXt101-32x4d

  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.3.1
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.06 is based on NVIDIA CUDA 11.3.1, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.06 is based on 1.9.0a0+c3d40fd
- Ubuntu 20.04 with May 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Starting in 21.06, PyProf will no longer be included in the NVIDIA PyTorch container. To profile models in PyTorch, please use NVIDIA Deep Learning Profiler (DLProf). DLProf can help data scientists, engineers and researchers understand and improve performance of their models with visualization via DLProf Viewer in the web browser, or by analyzing text reports. DL Prof is available on NGC or a Python PIP wheel installation.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.06	20.04	NVIDIA CUDA 11.3.1	1.9.0a0+c3d40fd	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.9.0a0+2ecb2c7	
21.04				
21.03		NVIDIA CUDA 11.2.1	1.9.0a0+df837d0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Known performance regressions in 21.06 vs. 21.05:
  - On Turing and NVIDIA Ampere Architecture GPUs:
    - Up to 15% performance drop for GNMT training
  - On Volta:
    - Up to 20% performance drop for Tacotron training.
- Manual synchronization is required in CUDA graphs workloads between graph replays.
- The PyTorch container includes a version of Django with a known vulnerability that was discovered late in our QA process. See CVE-2021-31542 for details. This will be fixed in the next release.
- The PyTorch container includes a version of Pillow with known vulnerabilities discovered late in our QA process. See CVE-2021-25287, CVE-2021-28676, CVE-2021-28677, and CVE-2021-25288 for details. This will be fixed in the next release.

Chapter 6. PyTorch Release 21.05

The NVIDIA container image for PyTorch, release 21.05, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.3.0
- cuBLAS 11.5.1.101
- NVIDIA cuDNN 8.2.0.51
- NVIDIA NCCL 2.9.8 (optimized for NVLink™)
- APEX
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.0
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.0
- Nsight Systems 2021.1.3.14
- TensorRT 7.2.3.4
- TensorBoard 1.15.5
- DALI 1.0.0
- MAGMA 2.5.2
- DLProf 1.1.0
- PyProf r21.05
- Tensor Core optimized examples:
    - ResNeXt101-32x4d

- SE-ResNext
- TransformerXL
- Jasper
- BERT
- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2

- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.3.1
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.05 is based on NVIDIA CUDA 11.3.0, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.05 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.05 is based on 1.9.0a0+2ecb2c7
- Ubuntu 20.04 with April 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Starting in 21.06, PyProf will no longer be included in the NVIDIA PyTorch container. To profile models in PyTorch, please use NVIDIA Deep Learning Profiler (DLProf). DLProf can help data scientists, engineers and researchers understand and improve performance of their models with visualization via DLProf Viewer in the web browser, or by analyzing text reports. DL Prof is available on NGC or a Python PIP wheel installation.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.05	20.04	NVIDIA CUDA 11.3.0	1.9.0a0+2ecb2c7	TensorRT 7.2.3.4
21.04				
21.03		NVIDIA CUDA 11.2.1	1.9.0a0+df837d0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12			1.4.0a0+174e1ba	TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
19.11				
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular

ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.

- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the

ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Known performance regressions in 21.05 vs. 21.04:
  - On NVIDIA Ampere Architecture GPUs:
    - Up to 17% performance drop for VGG16 training
- Manual synchronization is required in CUDA graphs workloads between graph replays.
- The DLProf TensorBoard plugin included with the 21.04 and 21.05 releases is an incorrect version with respect to the DLProf command line tool included in those releases. To correct this, use the following command:

```
$ pip install --index-urlhttps://developer.download.nvidia.com/compute/redist
 nvidia_tensorboard_plugin_dlprof==1.1.0
```

Chapter 7. PyTorch Release 21.04

The NVIDIA container image for PyTorch, release 21.04, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.3.0
- cuBLAS 11.5.1.101
- NVIDIA cuDNN 8.2.0.41
- NVIDIA NCCL 2.9.6 (optimized for NVLink™)
- APEX
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.0
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.18
- Nsight Systems 2021.1.3.14
- TensorRT 7.2.3.4
- TensorBoard 1.15.5
- DALI 1.0.0
- MAGMA 2.5.2
- DLProf 1.1.0
- PyProf r21.04
- Tensor Core optimized examples:
    - ResNeXt101-32x4d

- SE-ResNext
- TransformerXL
- Jasper
- BERT
- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2

- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.3.1
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.04 is based on NVIDIA CUDA 11.3.0, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.04 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.04 is based on 1.9.0a0+2ecb2c7
- Experimental release of the nvfuser backend for scripted models. Users can enable it using the context manager: `with.torch.jit.fuser("fuser2"):`

- Ubuntu 20.04 with March 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.04	20.04	NVIDIA CUDA 11.3.0	1.9.0a0+2ecb2c7	TensorRT 7.2.3.4
21.03		NVIDIA CUDA 11.2.1	1.9.0a0+df837d0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.

- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Known performance regressions in 21.04 vs. 21.03:
  - On NVIDIA Ampere Architecture GPUs:
    - Up to 38% performance drop for FastPitch training
    - Up to 30% performance drop in MaskRCNN training
  - On Turing:
    - Up to 20% performance drop in MaskRCNN training
    - Up to 15% performance drop in VGG16 training
- Manual synchronization is required in CUDA graphs workloads between graph replays.
- The DLProf TensorBoard plugin included with the 21.04 release is an incorrect version with respect to the DLProf command line tool included in those releases. To correct this, use the following command:

```
$ pip install --index-urlhttps://developer.download.nvidia.com/compute/redist
 nvidia_tensorboard_plugin_dlprof==1.1.0
```

Chapter 8. PyTorch Release 21.03

The NVIDIA container image for PyTorch, release 21.03, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.1026
- NVIDIA cuDNN 8.1.1
- NVIDIA NCCL 2.8.4 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 4.0.5
- TensorBoard 1.15.5
- Nsight Compute 2020.3.1.0
- Nsight Systems 2020.4.3.7
- TensorRT 7.2.2.3
- DALI 0.31.0
- MAGMA 2.5.2
- DLProf 1.0.0
- PyProf r21.03
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.0
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.03 is based on NVIDIA CUDA 11.2.1, which requires NVIDIA Driver release 460.32.03 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51(or later R450). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.03 is based on 1.9.0a0+df837d0
- NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.1026
- The latest version of NVIDIA cuDNN 8.1.1
- The latest version of DALI 0.31.0
- The latest version of DLProf 1.0.0
- The latest version of PyProf r21.03
- Ubuntu 20.04 with February 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.03	20.04	NVIDIA CUDA 11.2.1	1.9.0a0+df837d0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through	TensorRT 5.1.5

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
			commit 9130ab38 from July 31, 2019 as well as a cherry-picked	

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements

introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative

Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Known performance regressions in 21.03 vs. 21.02:

- On A102 (e.g. A40):
  - Up to 28% performance drop for BERT large inference and pre-training
  - Up to 33% performance drop for Transformer training
- On TU102 (e.g. RTX6000):
  - Up to 18% performance drop for inference on ResNet-like models
  - Up to 25% performance drop for SSD inference
  - Up to 15% performance drop for WaveGlow

Chapter 9. PyTorch Release 21.02

The NVIDIA container image for PyTorch, release 21.02, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.2.0 including cuBLAS 11.3.1
- NVIDIA cuDNN 8.1.0
- NVIDIA NCCL 2.8.4 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 4.0.5
- TensorBoard 1.15.5
- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- TensorRT 7.2.2.3+cuda11.1.0.024
- DALI 0.29.0
- MAGMA 2.5.2
- DLProf 0.19.0
- PyProf r21.02
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.0
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 21.02 is based on NVIDIA CUDA 11.2.0, which requires NVIDIA Driver release 460.27.04 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51(or later R450). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 21.02 is based on 1.8.0a0+52ea372
- NVIDIA CUDA 11.2.0 including cuBLAS 11.3.1
- The latest version of NVIDIA cuDNN 8.0.5
- The latest version of NVIDIA NCCL 2.8.4
- The latest version of Nsight Compute 2020.3.0.18
- The latest version of Nsight Systems 2020.4.3.7
- The latest version of TensorRT 7.2.2.3+cuda11.1.0.024

- The latest version of DALI 0.29
- The latest version of DLProf 0.19.0
- The latest version of PyProf r21.02
- Ubuntu 20.04 with January 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
21.02	20.04	NVIDIA CUDA 11.2.0	1.8.0a0+52ea372	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.

- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Known performance regressions in 21.02 vs. 20.12 for:
  - SSD-Resnet-50 training on Ampere up to 8%
  - ResNext101 inference on Volta and Ampere up to 24%
  - Se-ResNeXt101 inference on Ampere up to 17%
  - SSD inference on Volta up to 17%

Chapter 10. PyTorch Release 21.01

The NVIDIA container image release for PyTorch 21.01 has been canceled. The next release will be the 21.02 release which is expected to be released at the end of February.

Chapter 11. PyTorch Release 20.12

The NVIDIA container image for PyTorch, release 20.12, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8 environment
- NVIDIA CUDA 11.1.1 including cuBLAS 11.3.0
- NVIDIA cuDNN 8.0.5
- NVIDIA NCCL 2.8.3 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 4.0.5
- TensorBoard 1.15.0+nv20.11
- Nsight Compute 2020.2.1.8
- Nsight Systems 2020.3.4.32
- TensorRT 7.2.2
- DALI 0.28.0
- MAGMA 2.5.2
- DLProf 0.18.0
- PyProf r20.12
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT

    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1
    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.0
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.12 is based on NVIDIA CUDA 11.1.1, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.12 is based on 1.8.0a0+1606899
- NVIDIA CUDA 11.1.1 including cuBLAS 11.3.0
- The latest version of NVIDIA cuDNN 8.0.5
- The latest version of NVIDIA NCCL 2.8.3
- The latest version of Nsight Compute 2020.2.1.8
- The latest version of TensorRT 7.2.2
- The latest version of DALI 0.28
- The latest version of DLProf 0.18.0

- The latest version of PyProf r20.12
- Ubuntu 20.04 with November 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.12	20.04	NVIDIA CUDA 11.1.1	1.8.0a0+1606899	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31,	TensorRT 5.1.5

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
			2019 as well as a cherry-picked	

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase

published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known performance regressions in 20.12 vs. 20.11 for:
  - MaskR-CNN training up to 15%
  - Transformer-XL inference of approx. 10%
  - Tacotron2+Waveglow inference up to 50%
  - FastPitch inference and training up to 15%

Chapter 12. PyTorch Release 20.11

The NVIDIA container image for PyTorch, release 20.11, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.8.2 (optimized for NVLink™)
- APEX
- MLNX_OFED 5.1
- OpenMPI 4.0.5
- TensorBoard 1.15.0+nv20.11
- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- TensorRT 7.2.1
- DALI 0.27.0
- MAGMA 2.5.2
- DLProf 0.17.0
- PyProf r20.11
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.0
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 20.11 is based on NVIDIA CUDA 11.1.0, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.11 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.11 is based on 1.8.0a0+17f8c32
- The latest version of NVIDIA NCCL 2.8.2
- The latest version of DALI 0.27
- The latest version of DLProf 0.17.0
- The latest version of PyProf r20.11
- Ubuntu 18.04 with October 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0a0+17f8c32	TensorRT 7.2.1
20.10			1.7.0a0+7036e91	
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The PyProf version string in the 20.11 container is truncated to `3.` and does not display the full version string as `3.6.0`.
- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known performance regressions in 20.11 vs. 20.09 for:
  - MaskR-CNN training up to 9%
  - Transformer mixed-precision training up to 11%
  - Transformer-XL training and inference of approx. 10%
  - ResNet and ResNext inference up to 17%
- Known performance regressions in 20.11 vs. 2010 for:
  - GNMT mixed-precision training and inference up to 20%

Chapter 13. PyTorch Release 20.10

The NVIDIA container image for PyTorch, release 20.10, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 4.0.5
- TensorBoard 1.15.0+nv
- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- TensorRT 7.2.1
- DALI 0.26.0
- MAGMA 2.5.2
- DLProf 0.16.0
- PyProf r20.10
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.0
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 20.10 is based on NVIDIA CUDA 11.1.0, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.10 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, Turing, and NVIDIA Ampere GPU architecture families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.10 is based on 1.7.0a0+7036e91
- The latest version of NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- The latest version of NVIDIA cuDNN 8.0.4
- The latest version of TensorRT 7.2.1
- The latest version of OpenMPI 4.0.5
- The latest version of Nsight Compute 2020.2.0.18
- The latest version of Nsight Systems 2020.3.4.32
- The latest version of DALI 0.26

- The latest version of DLProf 0.16.0
- The latest version of PyProf 3.5.0
- Ubuntu 18.04 with September 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.10	18.04	NVIDIA CUDA 11.1.0	1.7.0a0+7036e91	TensorRT 7.2.1
20.09		NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31,	TensorRT 5.1.5

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
			2019 as well as a cherry-picked	

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase

published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The 20.10 container ships with libfreetype6 2.8.1-2ubuntu2.1, which is vulnerable to CVE-2020-15999. Use

  ```
  $ apt-get update
  $ apt --only-upgrade install libfreetype6
  ```

  to apply the patch to fix this issue, or alternatively, purge this package from the container.
- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known performance regressions in 20.10 vs. 20.09 for:
  - Bert mixed-precision training up to 13%
  - MaskR-CNN training up to 9%
  - Transformer mixed-precision training up to 14%
  - Transformer-XL training and inference of approx. 10%
  - ResNet and ResNext inference up to 17%
  - FastPitch FP32 training up to 13%

Chapter 14. PyTorch Release 20.09

The NVIDIA container image for PyTorch, release 20.09, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.6
- TensorBoard 1.15.0+nv
- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- TensorRT 7.1.3
- DALI 0.25.1
- MAGMA 2.5.2
- DLProf 0.15.0
- PyProf r20.09
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT

    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1
    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.0
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.09 is based on NVIDIA CUDA 11.0.3, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.09 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.09 is based on 1.7.0a0+8deb4fe with minor cherry-picked bug fixes
- The latest version of NVIDIA cuDNN 8.0.4
- The latest version of DALI 0.25.1
- The latest version of DLProf 0.15.0
- The latest version of PyProf r20.09
- Ubuntu 18.04 with August 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.09	18.04	NVIDIA CUDA 11.0.3	1.7.0a0+8deb4fe	TensorRT 7.1.3
20.08			1.7.0a0+6392713	
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models

without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known performance regressions in 20.08 vs. 20.07 for ResNet- and ResNext-like models on all architectures due to a temporal workaround in the dispatching mechanism (commit 0494e0a) up to 21%.
- Known NVIDIA Ampere GPU architecture performance regressions for FastPitch training in FP32 in 20.09 vs. 20.08 up to 10%.

Chapter 15. PyTorch Release 20.08

The NVIDIA container image for PyTorch, release 20.08, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- NVIDIA cuDNN 8.0.2
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.6
- TensorBoard 1.15.0+nv
- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- TensorRT 7.1.3
- DALI 0.24
- MAGMA 2.5.2
- DLProf 0.14.0
- PyProf r20.08
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1
    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.0
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.08 is based on NVIDIA CUDA 11.0.3, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.08 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.08 is based on 1.7.0a0+6392713 with minor cherry-picked bug fixes
- The latest version of NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- The latest version of NVIDIA NCCL 2.7.8
- The latest version of NVIDIA cuDNN 8.0.2
- The latest version of Nsight Compute 2020.1.2.4
- The latest version of Nsight Systems 2020.3.2.6
- The latest version of DALI 0.24

- The latest version of DLProf 0.14.0
- The latest version of PyProf r20.08
- Ubuntu 18.04 with July 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.08	18.04	NVIDIA CUDA 11.0.3	1.7.0a0+6392713	TensorRT 7.1.3
20.07		NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known performance regressions in 20.08 vs. 20.07 for ResNet- and ResNext-like models on all architectures due to a temporal workaround in the dispatching mechanism (commit 0494e0a) up to 18%.
- Known Turing performance regressions for FastPitch and WaveGlow inference in 20.08 vs. 20.07 up to 75%.

Chapter 16. PyTorch Release 20.07

The NVIDIA container image for PyTorch, release 20.07, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.0.194 including cuBLAS 11.1.0
- NVIDIA cuDNN 8.0.1
- NVIDIA NCCL 2.7.6 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.6
- TensorBoard 1.15.0+nv
- Nsight Compute 2020.1.1.8
- Nsight Systems 2020.3.2.6
- TensorRT 7.1.3
- DALI 0.23
- MAGMA 2.5.2
- DLProf 0.13.0
- PyProf r20.07
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

- Mask R-CNN
- Tacotron 2 and WaveGlow v1.1
- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.0
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 20.07 is based on NVIDIA CUDA 11.0.194, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.07 is based on 1.6.0a0+9907a3e with minor cherry-picked bug fixes
- The latest version of NVIDIA CUDA 11.0.194 including cuBLAS 11.1.0
- The latest version of NVIDIA NCCL 2.7.6
- The latest version of NVIDIA cuDNN 8.0.1
- The latest version of Nsight Compute 2020.1.1.8
- The latest version of Nsight Systems 2020.3.2.6
- The latest version of TensorRT 7.1.3

- The latest version of OpenMPI 3.1.6
- Ubuntu 18.04 with June 2020 updates
- Latest version of DALI 0.23

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.07	18.04	NVIDIA CUDA 11.0.194	1.6.0a0+9907a3e	TensorRT 7.1.3
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models

without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 5% performance drop on Transformer-XL mixed precision training in the 20.07 container compared to 19.11. Disabling the profiling executor at the beginning of your script might reduce this effect via:
```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```
- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known Turing performance regressions in 20.07 vs. 20.03 container:
  - Up to 10% performance drop on InceptionV3 for mixed precision training
  - Up to 10% performance drop on MaskNCF FP32 training
  - Up to 25% performance drop on MaskRCNN FP32 training
  - Up to 10% performance drop on ResNet50 for mixed precision training.
- Known Volta performance regressions in 20.07 vs. 20.03 container:
  - Up to 30% performance drop on WaveGlow for FP32 training
  - Up to 11% performance drop on ResNet101 and ResNet152 mixed precision training
  - Up to 10% performance drop on full FP16 VGG16 training
- Known Pascal performance regressions in 20.07 vs. 20.03 container:
  - Up to 19% performance drop on MaskRCNN for FP32 training
- When FFT Tiled algo are used with 3D convolution, an intermittent silent failure might happen due to dependency on the order of the stream execution. In some cases this might be manifested as NaNs in the output and we recommend to disable cuDNN via `torch.backends.cudnn.enabled = False`.
- Channels-last memory format is experimental in the 20.07 container. Potential convergence issues for ResNet variants are being investigated. On NVIDIA Ampere architecture based GPUs unexpected NaN values due to a race condition in a cuDNN kernel might be observed. We recommend to use the default memory format in case you run into these issues.

Chapter 17. PyTorch Release 20.06

The NVIDIA container image for PyTorch, release 20.06, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 11.0.167 including cuBLAS 11.1.0
- NVIDIA cuDNN 8.0.1
- NVIDIA NCCL 2.7.5 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.6
- TensorBoard 1.15.0+nv
- Nsight Compute 2020.1.0.33
- Nsight Systems 2020.2.5.8
- TensorRT 7.1.2
- DALI 0.22
- MAGMA 2.5.2
- DLProf 0.12.0
- PyProf r20.06
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT

   - Mask R-CNN
   - Tacotron 2 and WaveGlow v1.1
   - SSD300 v1.1
   - Neural Collaborative Filtering (NCF)
   - ResNet50 v1.5
   - GNMT v2
- Jupyter and JupyterLab:
   - Jupyter Client 6.0.0
   - Jupyter Core 4.6.1
   - Jupyter Notebook 6.0.3
   - JupyterLab 1.2.14
   - JupyterLab Server 1.0.6
   - Jupyter-TensorBoard

Driver Requirements

Release 20.06 is based on NVIDIA CUDA 11.0.167, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.06 is based on 1.6.0a0+9907a3e
- The latest version of NVIDIA CUDA 11.0.167 including cuBLAS 11.1.0
- The latest version of NVIDIA NCCL 2.7.5
- The latest version of NVIDIA cuDNN 8.0.1
- The latest version of Nsight Compute 2020.1.0.33
- The latest version of Nsight Systems 2020.2.5.8
- The latest version of TensorRT 7.1.2
- The latest version of OpenMPI 3.1.6

- Ubuntu 18.04 with May 2020 updates
- Latest version of DALI 0.22
- Native Automatic Mixed Precision (torch.cuda.amp). See official API documentation and examples. torch.cuda.amp is intended as the future-proof replacement for Apex AMP, and offers a number of advantages.
- Integrated latest NVIDIA Deep Learning SDK to support NVIDIA A100 using CUDA 11 and cuDNN 8
- Various bug fixes for channels-last layout optimization. Note that this layout is still in experimental form. See Known Issues below.
- Performance improvements for various torch.distribution methods by switching to the TensorIterator implementation
- Default TF32 support for Ampere-based GPUs

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.06	18.04	NVIDIA CUDA 11.0.167	1.6.0a0+9907a3e	TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12			1.4.0a0+174e1ba	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38	TensorRT 5.1.5

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
			from July 31, 2019 as well as a cherry-picked	

Automatic Mixed Precision (AMP)

Automatic Mixed Precision (AMP) for PyTorch is available in this container through the native implementation as well as a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Apex AMP is included to support models that currently rely on it, but torch.cuda.amp is the future-proof alternative, and offers a number of advantages over Apex AMP.

Guidance and examples demonstrating torch.cuda.amp can be found here.Apex AMP examples can be found here.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending

to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative

Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 5% performance drop on Transformer-XL mixed precision training in the 20.01 container compared to 19.11. Disabling the profiling executor at the beginning of your script might reduce this effect via:
```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```
- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- Known Turing performance regressions in 20.06 vs. 20.03 container:
  - Up to 10% performance drop on InceptionV3 for mixed precision training
  - Up to 10% performance drop on MaskNCF FP32 training
  - Up to 25% performance drop on MaskRCNN FP32 training
  - Up to 10% performance drop on ResNet50 for mixed precision training.
- Known Volta performance regressions in 20.06 vs. 20.03 container:
  - Up to 30% performance drop on WaveGlow for FP32 training
  - Up to 11% performance drop on ResNet101 and ResNet152 mixed precision training
- Known Pascal performance regressions in 20.06 vs. 20.03 container:
  - Up to 19% performance drop on MaskRCNN for FP32 training
- When FFT Tiled algo are used with 3D convolution, an intermittent silent failure might happen due to dependency on the order of the stream execution. In some cases this might be manifested as NaNs in the output and we recommend to disable cuDNN via `torch.backends.cudnn.enabled = False`.
- Channels-last memory format is experimental in the 20.06 container. Potential convergence issues for ResNet variants are being investigated. On NVIDIA Ampere architecture based GPUs unexpected NaN values due to a race condition in a cuDNN kernel might be observed. We recommend to use the default memory format in case you run into these issues.

Chapter 18. PyTorch Release 20.03

The NVIDIA container image for PyTorch, release 20.03, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.6.3 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.1.0
- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- TensorRT 7.0.0
- DALI 0.19.0
- MAGMA 2.5.2
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT
    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1

- SSD300 v1.1
- Neural Collaborative Filtering (NCF)
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 20.03 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.03 is based on 1.5.0a0+8f84ded
- Latest version of DALI 0.19.0
- Performance improvements for elementwise operations
- Performance improvements for per-channel quantization
- Relaxation of cudnn batchnorm input shape requirements
- Ubuntu 18.04 with February 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.03	18.04 16.04	NVIDIA CUDA 10.2.89	1.5.0a0+8f84ded	TensorRT 7.0.0
20.02			1.5.0a0+3bbb36e	
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 5% performance drop on Transformer-XL mixed precision training in the 20.01 container compared to 19.11. Disabling the profiling executor at the beginning of your script might reduce this effect via:

```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.

Chapter 19. PyTorch Release 20.02

The NVIDIA container image for PyTorch, release 20.02, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.1.0
- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- TensorRT 7.0.0
- DALI 0.18.0 Beta
- MAGMA 2.5.2
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1

    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.0.4
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.02 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.02 is based on PyTorch 1.4.0a0+a5b4d78 with a fix for wrong results in LU factorization using MAGMA<=2.5.1.
- Latest version of DALI 0.18.0 Beta
- Latest version of Nsight Systems 2020.1.1
- Latest version of Jupyter Notebook 6.0.3
- Ubuntu 18.04 with January 2020 updates
- Initial support for channel-last layout for convolutions
- Support for loop unrolling and vectorized loads and stores in TensorIterator
- Support for input activations with more than 2^{31} values

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- Transformer has been removed.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.02	18.04 16.04	NVIDIA CUDA 10.2.89	1.5.0a0+3bbb36e	TensorRT 7.0.0
20.01			1.4.0a0+a5b4d78	
19.12 19.11			1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 5% performance drop on Transformer-XL mixed precision training in the 20.01 container compared to 19.11. Disabling the profiling executor at the beginning of your script might reduce this effect via:

```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.

Chapter 20. PyTorch Release 20.01

The NVIDIA container image for PyTorch, release 20.01, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.1.0
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- TensorRT 7.0.0
- DALI 0.17.0 Beta
- MAGMA 2.5.2
- Tensor Core optimized examples:
    - ResNeXt101-32x4d
    - SE-ResNext
    - TransformerXL
    - Jasper
    - BERT
    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1

    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - Transformer
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.2
    - JupyterLab 1.0.4
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.01 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.01 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 20.01 is based on PyTorch 1.4.0a0+a5b4d78 with a fix for wrong results in LU factorization using MAGMA<=2.5.1.
- Latest version of TensorRT 7.0.0
- Latest version of DALI 0.17.0 Beta
- Latest version of MAGMA 2.5.2
- Ubuntu 18.04 with December 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
20.01	18.04	NVIDIA CUDA 10.2.89	1.4.0a0+a5b4d78	TensorRT 7.0.0
19.12 19.11	16.04		1.4.0a0+174e1ba	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.3.0a0+24ae9b5	
19.09			1.2.0	
19.08			1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked	TensorRT 5.1.5

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 5% performance drop on Transformer-XL mixed precision training in the 20.01 container compared to 19.11. Disabling the profiling executor at the beginning of your script might reduce this effect via:

```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```

- A workaround for the WaveGlow training regression from our past containers is to use a fake batch dimension when calculating the log determinant via `torch.logdet(W.unsqueeze(0).float()).squeeze()` as is done in this release.
- The mixed-precision recipe for Transformer training might create unexpectedly NaN outputs. We recommend using FP32 or AMP with `opt_level='O0'` with the 20.01 container.

Chapter 21. PyTorch Release 19.12

The NVIDIA container image for PyTorch, release 19.12, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.1.0
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- TensorRT 6.0.1
- DALI 0.16.0 Beta
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1

- Neural Collaborative Filtering (NCF)
- Transformer
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.2
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 19.12 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.30. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.12 is based on PyTorch 1.4.0a0+a5b4d78.
- Latest version of TensorBoard 2.1.0
- Latest version of DALI 0.16.0 Beta
- Latest version of Nsight Systems 2019.6.1
- Latest version of Jupyter Notebook 6.0.2
- Ubuntu 18.04 with November 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper

model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training

speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is up to 70% performance drop on Waveglow training in the 19.12 container compared to 19.10; due to a magma-CUDA bug recreating cuBLAS handles. You can minimize this regression by setting the `CUBLAS_WORKSPACE_CONFIG=:16:8` environment variable before running the Waveglow code.
- There is up to 20% performance drop for SSD training on Pascal GPUs in the 19.12 container compared to 19.11.
- There is up to 8% performance drop on BERT Large mixed-precision training in the 19.12 container compared to 19.10.
- The mixed-precision recipe for Transformer training might create unexpectedly NaN outputs. We recommend to use FP32 or AMP with `opt_level='O0'` with the 19.12 container.

Chapter 22. PyTorch Release 19.11

The NVIDIA container image for PyTorch, release 19.11, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.0.1
- Nsight Compute 2019.5.0
- Nsight Systems 2019.5.2
- TensorRT 6.0.1
- DALI 0.15.0 Beta
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1

- Neural Collaborative Filtering (NCF)
- Transformer
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook 5.7.8
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 19.11 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.30. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410 or 418.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.11 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.11 is based on PyTorch 1.4.0a0+174e1ba with cherry-picked fixes for TensorIterator, LayNerNorm as well as NCCL 2.5.
- Latest version of NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- Latest version of TensorBoard 2.0.1
- Latest version of NVIDIA cuDNN 7.6.5
- Latest version of NVIDIA NCCL 2.5.6
- Latest version of Nsight Compute 2019.5.0
- Latest version of Nsight Systems 2019.5.2
- Latest version of DALI 0.15.0 Beta
- Latest versions of Jupyter Client 5.3.4 and Jupyter Core 4.6.1

- Added a TransformerXL Tensor Core optimized example
- Ubuntu 18.04 with October 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.

- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.

- TransformerXL model. Transformer-XL is a transformer-based language model with a segment-level recurrence and a novel relative positional encoding. Enhancements introduced in Transformer-XL help capture better long-term dependencies by attending to tokens from multiple previous segments. Our implementation is based on the codebase

published by the authors of the Transformer-XL paper. Our implementation uses modified model architecture hyperparameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores. his model script is available on GitHub

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative

Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The performance of Mask R-CNN in FP32 precision is up to 14% slower in the 19.11 container compared to the 19.06 release. For best performance on Mask R-CNN, it is recommended to use automatic mixed precision training. This can easily be done using the float16 option with the MaskRCNN example included in this container.
- There is up to 66% performance drop on WaveGlow training in the 19.11 container compared to 19.10.
- The mixed-precision recipe for Transformer training might create unexpectedly NaN outputs. We recommend to use FP32 or AMP with `opt_level='O0'` with the 19.11 container.
- There is a 3-15% performance drop on Tacotron2 inference in the 19.11 container compared to 19.09.

Chapter 23. PyTorch Release 19.10

The NVIDIA container image for PyTorch, release 19.10, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.4
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 2.0.0
- Nsight Compute 2019.4.0
- Nsight Systems 2019.5.1
- TensorRT 6.0.1
- DALI 0.14.0 Beta
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)

- Transformer
- ResNet50 v1.5
- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.3
  - Jupyter Core 4.5.0
  - Jupyter Notebook 5.7.8
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 19.10 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.10 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.10 is based on PyTorch 1.3.0a0+24ae9b5.
- Latest version of NVIDIA cuDNN 7.6.4
- Latest version of TensorBoard 2.0.0
- Latest version of DALI 0.14.0 Beta
- Latest version of Nsight Systems 2019.5.1
- Latest version of Jupyter Client 5.3.3
- Added the ResNeXt101-32x4d and SE-ResNext Tensor Core optimized model examples.
- Ubuntu 18.04 with September 2019 updates

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- ResNeXt101-32x4d model. The ResNeXt101-32x4d is a model introduced in the Aggregated Residual Transformations for Deep Neural Networks paper. It is based on regular ResNet model, substituting 3x3 convolutions inside the bottleneck block for 3x3 grouped convolutions. This model script is available on GitHub.
- SE-ResNext model. The SE-ResNeXt101-32x4d is a ResNeXt101-32x4d model with added Squeeze-and-Excitation (SE) module introduced in the Squeeze-and-Excitation Networks paper. This model script is available on GitHub.
- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine

Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Performance of Mask R-CNN in FP32 precision is up to 20% slower in the 19.10 container compared to the 19.06 release. For best performance on Mask R-CNN, it is recommended to use automatic mixed precision training. This can easily be done using the float16 option with the MaskRCNN example included in this container.
- There is a 15-20% performance drop on WaveGlow inference in the 19.10 container compared to 19.08 using automatic mixed precision (AMP) with V100 compared to previous releases. To workaround this issue, install cuDNN 7.6.2 or use the 19.08 container.
- The mixed-precision recipe for Transformer training might create unexpectedly NaN outputs. We recommend to use FP32 or AMP with `opt_level='O0'` with the 19.10 container.
- There is a 3-15% performance drop on Tacotron2 inference in the 19.10 container compared to the previous release.

Chapter 24. PyTorch Release 19.09

The NVIDIA container image for PyTorch, release 19.09, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.3
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- APEX
- MLNX_OFED
- OpenMPI 3.1.4
- TensorBoard 1.14.0+nv
- Nsight Compute 2019.4.0
- Nsight Systems 2019.4.2
- TensorRT 6.0.1
- DALI 0.12.0 Beta
- Tensor Core optimized examples:
    - Jasper
    - BERT
    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1
    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - Transformer
    - ResNet50 v1.5

- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0
  - Jupyter Notebook 5.7.8
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 19.09 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.09 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.09 is based on PyTorch 1.2.0.
- Latest version of NVIDIA cuDNN 7.6.3
- Latest versions of Nsight Compute 2019.4.0 and Nsight Systems 2019.4.2
- Latest version of TensorRT 6.0.1
- Latest version of JupyterLab Server 1.0.6
- Ubuntu 18.04 with August 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in

the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Performance of Mask R-CNN in FP32 precision is up to 20% slower in the 19.07 container compared to the previous release. For best performance on Mask R-CNN, it is recommended to use automatic mixed precision training. This can easily be done using the float16 option with the MaskRCNN example included in this container.
- Due to recent changes on batch norm multiplier initialization (PyTorch commit: c60465873c5cf8f1a36da39f7875224d4c48d7ca), all batch norm multiplier is initialized as constant 1, instead of uniformly distributed between 0 and 1, as it was previously. This has caused accuracy issue for our TACOTRON2 model. If similar accuracy regression is observed during an update from 19.06 to 19.08, we recommend to re-initialize the batch norm multiplier using uniformed distribution. This could be done by passing your model to the following function:

```
def init_bn(module):
    if isinstance(module, torch.nn.modules.batchnorm._BatchNorm):
     if module.affine:
         module.weight.data.uniform_()
    for child in module.children():
     init_bn(child)
```

- There is a 34-60% performance drop on WaveGlow training in the 19.09 container on 16 GPU systems using mixed precision training compared to previous releases.
- There is a 15-20% performance drop on WaveGlow inference in the 19.09 container using automatic mixed precision (AMP) with V100 compared to previous releases. To workaround this issue, install cuDNN 7.6.2 or use the 19.08 container.
- Nsight Compute is currently located in `/opt/nvidia/nsight-compute/2019.4.0` , while Nsight Systems can be found in `/usr/local/cuda-10.1/NsightSystems-cli-2019.4.2/bin/nsys`.

Chapter 25. PyTorch Release 19.08

The NVIDIA container image for PyTorch, release 19.08, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.2
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- APEX
- MLNX_OFED +4.0
- OpenMPI 3.1.4
- TensorBoard 1.14.0+nv
- Nsight Compute 10.1.168
- Nsight Systems 2019.3.7.9
- TensorRT 5.1.5
- DALI 0.12.0 Beta
- Tensor Core optimized examples:
  - Jasper
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5

- GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0
  - Jupyter Notebook 5.7.8
  - JupyterLab 1.0.4
  - JupyterLab Server 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.08 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.87. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.08 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.08 is based on PyTorch 1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked performance fix 9462ca29.
- Latest version of NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- Latest version of NVIDIA cuDNN 7.6.2
- Latest version of NVIDIA NCCL 2.4.8
- Latest version of DALI 0.12.0 Beta
- Latest version of OpenMPI 3.1.4
- Latest version of Nsight Systems 2019.3.7.9
- Latest version of MLNX_OFED +4.0
- Added a Jasper Tensor Core model script example
- Ubuntu 18.04 with July 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- Jasper model. This repository provides an implementation of the Jasper model in PyTorch from the paper Jasper: An End-to-End Convolutional Neural Acoustic Model. The Jasper model is an end-to-end neural acoustic model for automatic speech recognition (ASR) that provides near state-of-the-art results on LibriSpeech among end-to-end ASR models without any external data. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- NCF model. The Neural Collaborative Filtering (NCF) model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Performance of Mask R-CNN in FP32 precision is up to 20% slower in the 19.07 container compared to the previous release. For best performance on Mask R-CNN, it is recommended to use automatic mixed precision training. This can easily be done using the float16 option with the MaskRCNN example included in this container.

- Due to recent changes on batch norm multiplier initialization (PyTorch commit: c60465873c5cf8f1a36da39f7875224d4c48d7ca), all batch norm multiplier is initialized as constant 1, instead of uniformly distributed between 0 and 1, as it was previously. This has caused accuracy issue for our TACOTRON2 model. If similar accuracy regression is observed during an update from 19.06 to 19.08, we recommend to re-initialize the batch norm multiplier using uniformed distribution. This could be done by passing your model to the following function:

```
def init_bn(module):
    if isinstance(module, torch.nn.modules.batchnorm._BatchNorm):
     if module.affine:
         module.weight.data.uniform_()
    for child in module.children():
     init_bn(child)
```

Chapter 26. PyTorch Release 19.07

The NVIDIA container image for PyTorch, release 19.07, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 18.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- NVIDIA cuDNN 7.6.1
- NVIDIA NCCL 2.4.7 (optimized for NVLink™)
- APEX
- MLNX_OFED +3.4
- OpenMPI 3.1.3
- TensorBoard 1.14.0+nv
- TensorRT 5.1.5
- DALI 0.11.0 Beta
- Tensor Core optimized examples:
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:

- Jupyter Client 5.3.1
- Jupyter Core 4.5.0
- Jupyter Notebook 5.7.8
- JupyterLab 1.0.2
- JupyterLab Server 1.0.0
- Jupyter-TensorBoard

Driver Requirements

Release 19.07 is based on NVIDIA CUDA 10.1.168, which requires NVIDIA Driver release 418.67. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.07 is based on PyTorch 1.2.0a0 including upstream commits up through commit f6aac41 from June 19, 2019.
- Latest version of NVIDIA cuDNN 7.6.1
- Latest version of MLNX_OFED +3.4
- Added TensorBoard 1.14.0+nv to the container.
- Latest versions of Jupyter Client 5.3.1, Jupyter Core 4.5.0, JupyterLab 1.0.2 and JupyterLab Server 1.0.0, including Jupyter-TensorBoard integration.
- Latest version of DALI 0.11.0 Beta
- Latest version of Ubuntu 18.04

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy.
- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy.
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper.
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model.
- NCF model. The Neural Collaborative Filtering (NCF) focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Performance of Mask R-CNN in FP32 precision is up to 20% slower in the 19.07 container compared to the previous release. For best performance on Mask R-CNN, it is recommended to use automatic mixed precision training. This can easily be done using the float16 option with the MaskRCNN example included in this container.
- Due to recent changes on batch norm multiplier initialization (PyTorch commit: c60465873c5cf8f1a36da39f7875224d4c48d7ca), all batch norm multiplier is initialized as constant 1, instead of uniformly distributed between 0 and 1, as it was previously. This has caused accuracy issue for our TACOTRON2 model. If similar accuracy regression is observed during an update from 19.06 to 19.07, we recommend to re-initialize the batch norm multiplier using uniformed distribution. This could be done by passing your model to the following function:

```
def init_bn(module):
    if isinstance(module, torch.nn.modules.batchnorm._BatchNorm):
     if module.affine:
         module.weight.data.uniform_()
    for child in module.children():
     init_bn(child)
```

Chapter 27. PyTorch Release 19.06

The NVIDIA container image for PyTorch, release 19.06, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- NVIDIA cuDNN 7.6.0
- NVIDIA NCCL 2.4.7 (optimized for NVLink™)
- APEX
- OpenMPI 3.1.3
- TensorRT 5.1.5
- DALI 0.10.0 Beta
- Tensor Core optimized examples:
  - BERT
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.2.4
  - Jupyter Core 4.4.0

- Jupyter Notebook 5.7.8
- JupyterLab 0.35.6
- JupyterLab Server 0.2.0

Driver Requirements

Release 19.06 is based on NVIDIA CUDA 10.1.168, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.06 is based on PyTorch 1.1.0commit 0885dd28 from May 28, 2019
- Added the BERT Tensor Core example
- Latest version of NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- Latest version of NVIDIA NCCL 2.4.7
- Latest version of DALI 0.10.0 Beta
- Latest version of JupyterLab 0.35.6
- Ubuntu 16.04 with May 2019 updates (see Announcements)

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on theBERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's implementation of BERT is an optimized version of the Hugging Face implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUs for faster training times while maintaining target accuracy.

- Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy.
- Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper.
- SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model.
- NCF model. The Neural Collaborative Filtering (NCF) focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- Transformer model. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- ResNet50 v1.5 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 3 lines of Python to an existing FP32 (default) script. Amp will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Announcements

In the next release, we will no longer support Ubuntu 16.04. Release 19.07 will instead support Ubuntu 18.04.

Known Issues

- There is a known issue when running certain tests in PyTorch 19.06 on systems with Skylake CPUs, such as DGX-2, that is due to OpenBLAS version 0.3.6. If you are impacted, run:

```
/opt/conda/bin/conda install openblas!=0.3.6
```

Chapter 28. PyTorch Release 19.05

The NVIDIA container image for PyTorch, release 19.05, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.1 Update 1 including cuBLAS 10.1 Update 1
- NVIDIA cuDNN 7.6.0
- NVIDIA NCCL 2.4.6 (optimized for NVLink™)
- APEX
- OpenMPI 3.1.3
- TensorRT 5.1.5
- DALI 0.9.1 Beta
- Tensor Core optimized examples:
    - Mask R-CNN
    - Tacotron 2 and WaveGlow v1.1
    - SSD300 v1.1
    - Neural Collaborative Filtering (NCF)
    - Transformer
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 5.2.4
    - Jupyter Core 4.4.0
    - JupyterLab 0.35.4

- JupyterLab Server 0.2.0

Driver Requirements

Release 19.05 is based on CUDA 10.1 Update 1, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.05 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.05 is based on PyTorch 1.0.1 commit 828a6a3b from March 31, 2019
- Latest version of NVIDIA CUDA 10.1 Update 1 including cuBLAS 10.1 Update 1
- Latest version of NVIDIA cuDNN 7.6.0
- Latest version of TensorRT 5.1.5
- Latest version of DALI 0.9.1 Beta
- Ubuntu 16.04 with April 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy.

- An implementation of the Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper.
- An implementation of the SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model.
- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- An implementation of the Transformer model architecture. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.
- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 2 lines of Python to an existing FP32 (default) script. AMP will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Persistent batch normalization kernels are enabled by default in this build. This will provide a performance boost to many networks, but in rare cases may cause a network to fail to train properly. We expect to address this in the 19.06 container.

Chapter 29. PyTorch Release 19.04

The NVIDIA container image for PyTorch, release 19.04, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.105 including cuBLAS 10.1.0.105
- NVIDIA cuDNN 7.5.0
- NVIDIA NCCL 2.4.6 (optimized for NVLink™)
- APEX
- OpenMPI 3.1.3
- TensorRT 5.1.2
- DALI 0.8.1 Beta
- Tensor Core optimized examples:
  - Mask R-CNN
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.2.4
  - Jupyter Core 4.4.0
  - JupyterLab 0.35.4

- JupyterLab Server 0.2.0

Driver Requirements

Release 19.04 is based on CUDA 10.1, which requires NVIDIA Driver release 418.xx.x+. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.04 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.04 is based on PyTorch 1.0.1 commit 9eb0f43 from March 28, 2019
- Latest version of NVIDIA NCCL 2.4.6
- Latest version of DALI 0.8.1 Beta
- Latest version of cuBLAS 10.1.0.105
- Added the Mask R-CNN Tensor Core example
- Ubuntu 16.04 with March 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the Mask R-CNN model. Mask R-CNN is a convolution based neural network for the task of object instance segmentation. The paper describing the model can be found here. NVIDIA's Mask R-CNN model is an optimized version of Facebook's implementation, leveraging mixed precision arithmetic using Tensor Cores on NVIDIA Tesla V100 GPUs for 1.3x faster training time while maintaining target accuracy.

- An implementation of the Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper.
- An implementation of the SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model.
- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- An implementation of the Transformer model architecture. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.
- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Automatic Mixed Precision (AMP)

NVIDIA's Automatic Mixed Precision (AMP) for PyTorch is available in this container through a preinstalled release of Apex. AMP enables users to try mixed precision training by adding only 2 lines of Python to an existing FP32 (default) script. AMP will choose an optimal set of operations to cast to FP16. FP16 operations require 2X reduced memory bandwidth (resulting in a 2X speedup for bandwidth-bound operations like most pointwise ops) and 2X reduced memory storage for intermediates (reducing the overall memory consumption of your model). Additionally, GEMMs and convolutions with FP16 inputs can run on Tensor Cores, which provide an 8X increase in computational throughput over FP32 arithmetic.

Comprehensive guidance and examples demonstrating AMP for PyTorch can be found in the documentation.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Persistent batch normalization kernels are enabled by default in this build. This will provide a performance boost to many networks, but in rare cases may cause a network to fail to train properly. We expect to address this in the 19.05 container.

Chapter 30. PyTorch Release 19.03

The NVIDIA container image for PyTorch, release 19.03, is available on NGC.

Contents of the PyTorch container

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.1.105 including cuBLAS 10.1.105
- NVIDIA cuDNN 7.5.0
- NVIDIA NCCL 2.4.3 (optimized for NVLink™)
- APEX
- OpenMPI 3.1.3
- TensorRT 5.1.2
- DALI 0.7 Beta
- Tensor Core optimized examples:
  - Tacotron 2 and WaveGlow v1.1
  - SSD300 v1.1
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5
  - GNMT v2
- Jupyter and JupyterLab:
  - Jupyter Client 5.2.4
  - Jupyter Core 4.4.0
  - JupyterLab 0.35.4
  - JupyterLab Server 0.2.0

Driver Requirements

Release 19.03 is based on CUDA 10.1, which requires NVIDIA Driver release 418.xx+. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.03 is based on PyTorch commit 81e025d from March 9th, 2019
- Latest version of NVIDIA CUDA 10.1.105 including cuBLAS 10.1.105
- Latest version of NVIDIA cuDNN 7.5.0
- Latest version of NVIDIA NCCL 2.4.3
- Latest version of DALI 0.7 Beta
- Latest version of TensorRT 5.1.2
- Added the Tacotron 2 and WaveGlow v1.1 and SSD300 v1.1 Tensor Core examples
- Ubuntu 16.04 with February 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the Tacotron 2 and WaveGlow v1.1 model. This text-to-speech (TTS) system is a combination of two neural network models: a modified Tacotron 2 model from the Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions paper and a flow-based neural network model from the WaveGlow: A Flow-based Generative Network for Speech Synthesis paper.

- An implementation of the SSD300 v1.1 model. The SSD300 v1.1 model is based on the SSD: Single Shot MultiBox Detector paper. The main difference between this model and the one described in the paper is in the backbone. Specifically, the VGG model is obsolete and is replaced by the ResNet50 model.
- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- An implementation of the Transformer model architecture. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.
- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

- Persistent batch normalization kernels have been disabled due to a known bug during validation. Batch normalization provides correct results and work as expected from users, however, this may cause up to 10% regression in time to solution performance on networks using batch normalization.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 31. PyTorch Release 19.02

The NVIDIA container image for PyTorch, release 19.02, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda™ default environment (`/opt/conda/lib/python3.6/site-packages/torch/`) in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.2
- NVIDIA Collective Communications Library (NCCL) 2.3.7 (optimized for NVLink™)
- APEX
- OpenMPI 3.1.3
- TensorRT 5.0.2
- DALI 0.6.1 Beta
- Tensor Core optimized examples:
    - Neural Collaborative Filtering (NCF)
    - Transformer
    - ResNet50 v1.5
    - GNMT v2
- Jupyter and JupyterLab:
    - Jupyter Client 5.2.4
    - Jupyter Core 4.4.0
    - JupyterLab 0.35.4
    - JupyterLab Server 0.2.0

Driver Requirements

Release 19.02 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.02 is based on PyTorch Version 1.1.0a0+c42431b.
- Latest version of DALI 0.6.1 Beta
- Added Jupyter and JupyterLab software in our packaged container.
- Latest version of jupyter_client 5.2.4
- Latest version of jupyter_core 4.4.0
- Ubuntu 16.04 with January 2019 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Coress by using the latest deep learning example networks for training. This container includes the following Tensor Core examples.

- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.
- An implementation of the Transformer model architecture. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.
- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

- Persistent batch normalization kernels have been disabled due to a known bug during validation. Batch normalization provides correct results and work as expected from users, however, this may cause up to 10% regression in time to solution performance on networks using batch normalization.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 32. PyTorch Release 19.01

The NVIDIA container image for PyTorch, release 19.01, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.2
- NCCL 2.3.7 (optimized for NVLink™)
- OpenMPI 3.1.3
- Caffe2
- TensorRT 5.0.2
- DALI 0.6 Beta
- Tensor Core optimized examples:
  - Neural Collaborative Filtering (NCF)
  - Transformer
  - ResNet50 v1.5
  - GNMT v2

Driver Requirements

Release 19.01 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.01 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 19.01 is based on PyTorch v1.0.0 with up-to-date features.
- Latest version of DALI 0.6 Beta
- Latest version of NVIDIA cuDNN 7.4.2
- Latest version of OpenMPI 3.1.3
- Added the Neural Collaborative Filtering (NCF) and Transformer Tensor Core examples.
- Ubuntu 16.04 with December 2018 updates

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training. This container includes the following Tensor Core examples.

- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model focuses on providing recommendations, also known as collaborative filtering; with implicit feedback. The training data for this model should contain binary information about whether a user interacted with a specific item. NCF was first described by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua in the Neural Collaborative Filtering paper.

- An implementation of the Transformer model architecture. The Transformer model is based on the optimized implementation in Facebook's Fairseq NLP Toolkit and is built on top of PyTorch. The original version in the Fairseq project was developed using Tensor Cores, which provides significant training speedup. Our implementation improves the performance and is tested on a DGX-1V 16GB.

- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.

- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

- Persistent batch normalization kernels have been disabled due to a known bug during validation. Batch normalization provides correct results and work as expected from users, however, this may cause up to 10% regression in time to solution performance on networks using batch normalization.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 33. PyTorch Release 18.12

The NVIDIA container image for PyTorch, release 18.12, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.1
- NCCL 2.3.7 (optimized for NVLink™)
- APEx
- OpenMPI 3.1.2
- Caffe2
- TensorRT 5.0.2
- DALI 0.5.0 Beta
- Tensor Core Optimized Examples:
    - ResNet50 v1.5
    - GNMT v2

Driver Requirements

Release 18.12 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 18.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.12 is based on PyTorch v0.4.1+ with up-to-date features from the PyTorch v1.0 preview (main branch up to PR 12303). PyTorch 0.4.1+ is released and included with this container.
- Performance improvement for PyTorch's native batch normalization.
- Mixed precision SoftMax enabling FP16 inputs, FP32 computations and FP32 outputs.
- Latest version of DALI 0.5.0 Beta.
- Ubuntu 16.04 with November 2018 updates

Tensor Core Examples

- An implementation of ResNet50. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.
- An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

Persistent batch normalization kernels have been disabled due to a known bug during validation. Batch normalization provides correct results and work as expected from users, however, this may cause up to 10% regression in time to solution performance on networks using batch normalization.

Chapter 34. PyTorch Release 18.11

The NVIDIA container image for PyTorch, release 18.11, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.1
- NCCL 2.3.7 (optimized for NVLink™)
- APEx
- OpenMPI 3.1.2
- Caffe2
- TensorRT 5.0.2
- DALI 0.4.1 Beta
- Tensor Core Optimized Examples:
  - ResNet50 v1.5
  - GNMT v2

Driver Requirements

Release 18.11 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.11 is based on PyTorch v0.4.1+ with up-to-date features from the PyTorch v1.0 preview (main branch up to PR 11834). PyTorch 0.4.1+ is released and included with this container.
- Latest version of NCCL 2.3.7.
- Latest version of NVIDIA cuDNN 7.4.1.
- Latest version of TensorRT 5.0.2
- Latest version of DALI 0.4.1 Beta.
- Ubuntu 16.04 with October 2018 updates

Tensor Core Examples

- An implementation of ResNet50. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.
- An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

There is a known bug when using persistent batch normalization kernels. If you are experiencing a drop in predictive power during testing and validation, the recommended workaround is to not add the `.eval()` flag on your model when doing testing or validation.

Chapter 35. PyTorch Release 18.10

The NVIDIA container image of PyTorch, release 18.10, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.0
- NCCL 2.3.6 (optimized for NVLink™)
- Caffe2
- APEx
- OpenMPI 3.1.2
- TensorRT 5.0.0 RC
- DALI 0.4 Beta
- Tensor Core Optimized Examples:
  - ResNet50 v1.5
  - GNMT v2

Driver Requirements

Release 18.10 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.10 is based on PyTorch v0.4.1+ with up-to-date features from the PyTorch v1.0 preview (main branch up to PR 11834). PyTorch 0.4.1+ is released and included with this container.
- When possible PyTorch will now automatically use cuDNN persistent RNN's providing improved speed for smaller RNN's.
- Improved multi-GPU performance in both PyTorch c10d and Apex's DDP.
- Faster weight norm with improved mixed-precision accuracy used through `torch.nn.utils.weight_norm`.
- Improved functionality of the `torch.jit.script` and `torch.jit.trace`preview features including better support for pointwise operations in fusion.
- Added support for a C++ only API (new PyTorch 1.0 preview feature).
- Dataloader may still throw a benign error when stopping iterations early, however, it is no longer preventing the process from ending.
- Latest version of DALI 0.4 Beta.
- Latest version of NCCL 2.3.6.
- Added support for OpenMPI 3.1.2
- Ubuntu 16.04 with September 2018 updates

Tensor Core Examples

- An implementation of ResNet50. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.
- An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

There are no new issues in this release.

Chapter 36. PyTorch Release 18.09

The NVIDIA container image of PyTorch, release 18.09, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.3.0
- NCCL 2.3.4 (optimized for NVLink™)
- Caffe2
- TensorRT 5.0.0 RC
- DALI 0.2 Beta
- Tensor Core Optimized Examples:
  - ResNet50 v1.5
  - GNMT v2

Driver Requirements

Release 18.09 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.09 is based on PyTorch 0.4.1+. PyTorch 0.4.1 is released and included with this container.

- Latest version of cuDNN 7.3.0.
- Latest version of CUDA 10.0.130 which includes support for DGX-2, Turing, and Jetson Xavier.
- Latest version of cuBLAS 10.0.130.
- Latest version of NCCL 2.3.4.
- Latest version of TensorRT 5.0.0 RC.

> **Note:** All 18.09 containers inherit TensorRT 5.0.0 RC from the base container, however, some containers may not use TensorRT if there is no support for TensorRT in the given framework.

- An implementation of ResNet50. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.
- Stream pool: PyTorch now uses per GPU stream pools behind the scenes. This means that CUDA streams are created when first used on a GPU and destroyed on exit. As a result, networks that use multiple streams may see the same stream used repeatedly in their profiles, and networks that retain streams for long periods may accidentally schedule parallelizable work to the same stream. It's recommended that streams be acquired, used, and released as needed.
- Reliability: Some cases where a dataloader could hang if shutdown during its iteration has been fixed.
- Fusion: Tensor and constant scalar operations, like `add(t, 1)`, and `chunk` operations are now fusable.
- Performance improvements: `dropout`, 1x1 convolutions for `NCHW`, and `weightnorm` should be faster in a majority of scenarios.
- Latest version of DALI 0.2 Beta
- Ubuntu 16.04 with August 2018 updates

Tensor Core Examples

- An implementation of ResNet50. The ResNet50 v1.5 model is a modified version of the original ResNet50 v1 model.
- An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

- The DALI integrated ResNet-50 samples in the 18.09 NGC TensorFlow and PyTorch containers may result in lower than expected performance results. We are working to address the issue in the next release.

- There is a chance that PyTorch will hang on exit when running multi-gpu training. This hang does not affect any results of the run; however, the process will have to be terminated manually.

Chapter 37. PyTorch Release 18.08

The NVIDIA container image of PyTorch, release 18.08, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.2.1
- NCCL 2.2.13 (optimized for NVLink™)
- Caffe2 0.8.1
- DALI 0.1.2 Beta
- Tensor Core Optimized Examples:
  - GNMT v2

Driver Requirements

Release 18.08 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.08 is based on PyTorch 0.4.1. PyTorch 0.4.1 is released and included with this container. See the release notes at https://github.com/pytorch/pytorch/releases for significant changes from PyTorch 0.4.
- Apex is now entirely Python for improved compatibility. Previous versions of Apex will not work with PyTorch 0.4.1 or newer versions.

- New ops in 18.08: `torch.pinverse`, `torch.unique`, `torch.erfc`, `torch.isinf`, `torch.isfinite`, `torch.reshape_as`.
- Support for cross-device gradient clipping.
- `torch.svd` and `torch.eig` in CUDA have been fixed, previously they could return incorrect results.
- An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.
- Latest version of cuDNN 7.2.1.
- Latest version of DALI 0.1.2 Beta.
- Added support for the Tensor Core Optimized Example: PyTorch GNMT model
- Ubuntu 16.04 with July 2018 updates

Usage

The `PYTHONPATH` environment variable in this container version has been updated to include all packages installed in the Conda environment and all PyTorch related packages. Users that rely on `PYTHONPATH` to point to local modules are advised to carefully check and set their `PYTHONPATH` variable in this container and moving forward.

Tensor Core Examples

An implementation of GNMT v2. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper.

Known Issues

The DALI integrated ResNet-50 samples in the 18.08 NGC TensorFlow and PyTorch containers may result in lower than expected performance results. We are working to address the issue in the next release.

Chapter 38. PyTorch Release 18.07

The NVIDIA container image of PyTorch, release 18.07, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.4
- NCCL 2.2.13 (optimized for NVLink™)
- Caffe2 0.8.1
- DALI 0.1 Beta

Driver Requirements

Release 18.07 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.07 is based on PyTorch 0.4.0 upstream main branch post commit cca2476.
- Clip grads can be used on a single tensor directly.
- The precision of MSELoss with half inputs has been improved.
- PyTorch's JIT (still in Alpha) now supports FP16 inputs and outputs, comparisons, the exp operator, and ReLU gates.
- Added support for DALI 0.1 Beta.
- Latest version of CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425.

- Ubuntu 16.04 with June 2018 updates

Known Issues

When importing Caffe2 after importing Torch, there is an issue which causes GPU support for Caffe2 to be disabled. For users affected by this bug, it is recommended to either use the PyTorch 18.06 or 18.08 container.

Chapter 39. PyTorch Release 18.06

The NVIDIA container image of PyTorch, release 18.06, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.4
- NCCL 2.2.13 (optimized for NVLink™)
- Caffe2 0.8.1

Driver Requirements

Release 18.06 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.06 is based on PyTorch 0.4.0 upstream main branch post commit 0e9613c.
- Improved data loader pipeline in the ImageNet example, see `/opt/pytorch/examples/imagenet` within the container.
- Data loader pipeline now uses `pillow-simd` and `jpeg-turbo`.
- Improved FP16 support, specifically, reductions like `sum()` are now more accurate when using FP16.
- Improved distributed performance, specifically, gradient communication can now overlap with gradient computation in `backwards()`.
- Compatibility changes, specifically, Magma 1 is no longer supported.

- Ubuntu 16.04 with May 2018 updates

Known Issues

There are no known issues in this release.

Chapter 40. PyTorch Release 18.05

The NVIDIA container image of PyTorch, release 18.05, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.2
- NCCL 2.1.15 (optimized for NVLink™)
- Caffe2 0.8.1

Driver Requirements

Release 18.05 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.05 is based on PyTorch 0.4.0.
- Includes Caffe2 0.8.1. For more information, see PyTorch and Caffe2 repos getting closer together.
- APEx, an extension providing utilities for FP16 and multi-gpu training. For more information, see APEx: A PyTorch Extension and APEx.
- Ubuntu 16.04 with April 2018 updates

Known Issues

- Some mixed-precision models might encounter a crash due to a new FP16 overflow check added in PyTorch. We have an upstream fix submitted with PR 7382 and should be resolved in a future container.
- There is a minor performance regression with the imagenet sample in `/opt/pytorch/examples/imagnet` for some network architectures on multi-gpu cases. This regression will be fixed in the next release.

Chapter 41. PyTorch Release 18.04

The NVIDIA container image of PyTorch, release 18.04, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.1
- NCCL 2.1.15 (optimized for NVLink™)

Driver Requirements

Release 18.04 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.04 is based on PyTorch 0.3.1.
- Incorporated all upstream changes from the PyTorch main branch, specifically up to and including commit 2f27c1b5.
- Latest version of NCCL 2.1.15
- Ubuntu 16.04 with March 2018 updates

Known Issues

Some mixed-precision models might encounter a crash due to a new FP16 overflow check added in PyTorch. We have an upstream fix submitted with PR 7382 and should be resolved in a future container.

Chapter 42. PyTorch Release 18.03

The NVIDIA container image of PyTorch, release 18.03, is available.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.1
- NCCL 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 18.03 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- PyTorch container image version 18.03 is based on PyTorch 0.3.0.
- Incorporated all upstream changes from the PyTorch main branch, specifically, PR 5327.
- Latest version of cuBLAS 9.0.333
- Latest version of cuDNN 7.1.1
- Ubuntu 16.04 with February 2018 updates

Known Issues

There are no known issues in this release.

Chapter 43. PyTorch Release 18.02

The NVIDIA container image of PyTorch, release 18.02, is available.

PyTorch container image version 18.02 is based on PyTorch 0.3.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 including:
  - CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.282 Patch 2 which is installed by default
  - cuBLAS 9.0.234 Patch 1 as a debian file. Installing Patch 1 by issuing the `dpkg -i /opt/cuda-cublas-9-0_9.0.234-1_amd64.deb` command is the workaround for the known issue described below.
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NCCL 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 18.02 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Improved multi-GPU performance on image networks shown in `/opt/pytorch/examples/imagenet`. You can run this example for multi-GPU by issuing the `python -m multiproc main.py` command.
- Latest version of cuBLAS
- Ubuntu 16.04 with January 2018 updates

Known Issues

cuBLAS 9.0.282 regresses RNN seq2seq FP16 performance for a small subset of input sizes. This issue should be fixed in the next update. As a workaround, install cuBLAS 9.0.234 Patch 1 by issuing the `dpkg -i /opt/cuda-cublas-9-0_9.0.234-1_amd64.deb` command.

Chapter 44. PyTorch Release 18.01

The NVIDIA container image of PyTorch, release 18.01, is available.

PyTorch container image version 18.01 is based on PyTorch 0.3.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py3.6` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04 including Python 3.6 environment
- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.282
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NCCL 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 18.01 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Latest version of cuBLAS
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with December 2017 updates

Known Issues

cuBLAS 9.0.282 regresses RNN seq2seq FP16 performance for a small subset of input sizes. As a workaround, revert back to the 11.12 container.

Chapter 45. PyTorch Release 17.12

The NVIDIA container image of PyTorch, release 17.12, is available.

PyTorch container image version 17.12 is based on PyTorch 0.2.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py35` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.234
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NCCL 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 17.12 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with November 2017 updates

Known Issues

There are no known issues in this release.

Chapter 46. PyTorch Release 17.11

The NVIDIA container image of PyTorch, release 17.11, is available.

PyTorch container image version 17.11 is based on PyTorch 0.2.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py35` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.234
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.4
- NCCL 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 17.11 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Tensor Op accelerated RNNs for Volta architecture
- Improved depthwise separable convolution performance
- Improved automatic differentiation engine latency
- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with October 2017 updates

Known Issues

There are no known issues in this release.

Chapter 47. PyTorch Release 17.10

The NVIDIA container image of PyTorch, release 17.10, is available.

PyTorch container image version 17.10 is based on PyTorch 0.2.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py35` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.3
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.5 (optimized for NVLink™)

Driver Requirements

Release 17.10 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with September 2017 updates

Known Issues

There are no known issues in this release.

Chapter 48. PyTorch Release 17.09

The NVIDIA container image of PyTorch, release 17.09, is available.

PyTorch container image version 17.09 is based on PyTorch 0.2.0.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in the `pytorch-py35` Conda™ environment in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.2
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.5 (optimized for NVLink™)

Driver Requirements

Release 17.09 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Supports Tensor Core operations for convolutions and GEMMs on Volta hardware
- The `examples` directory contains examples of ImageNet and LSTM training scripts that use FP16 data, as well as show how to train with FP16
- Matrix multiplication on FP16 inputs uses Tensor Core math when available
- A custom batch normalization layer is implemented to use cuDNN for batch normalization with FP16 inputs
- Latest version of CUDA
- Latest version of cuDNN with support for Tensor Core math when available
- Latest version of NCCL
- Ubuntu 16.04 with August 2017 updates

Known Issues

There are no known issues in this release.

Chapter 49. PyTorch Release 17.07

The NVIDIA container image of PyTorch, release 17.07, is available.

PyTorch container image version 17.07 is based on PyTorch 0.1.12.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed into the `/usr/local/[bin,share,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61.2 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) Patch 2
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.3 (optimized for NVLink™)

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Support for advanced tensor indexing
- Support multi-node or multi-process mode on the same node
- Support for double backward for most functions, including convolution
- Ubuntu 16.04 with June 2017 updates

Known Issues

There are no known issues in this release.

Chapter 50. PyTorch Release 17.06

The NVIDIA container image of PyTorch, release 17.06, is available.

PyTorch container image version 17.06 is based on PyTorch 0.1.12.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed into the `/usr/local/[bin,share,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Ubuntu 16.04 with May 2017 updates

Known Issues

The NCCL library version 1.6.1 included in this image, modifies the output buffers on all GPUs during in-place `ncclReduce()` operations, whereas normally only the "root" (target) device's output buffer should be modified. This is fixed in later versions of NCCL, as will be packaged in later versions of this image. As a workaround, either use `ncclAllReduce()`, which correctly modifies output buffers of all GPUs to the same values, or use out-of-place `ncclReduce()`, wherein the output buffer is distinct from the input buffer.

Chapter 51. PyTorch Release 17.05

The NVIDIA container image of PyTorch, release 17.05, is available.

PyTorch container image version 17.05 is based on PyTorch 0.1.12.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed into the `/usr/local/[bin,share,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Latest cuDNN release
- Ubuntu 16.04 with April 2017 updates

Known Issues

The NCCL library version 1.6.1 included in this image, modifies the output buffers on all GPUs during in-place `ncclReduce()` operations, whereas normally only the "root" (target) device's output buffer should be modified. This is fixed in later versions of NCCL, as will be packaged in later versions of this image. As a workaround, either use `ncclAllReduce()`, which correctly modifies output buffers of all GPUs to the same values, or use out-of-place `ncclReduce()`, wherein the output buffer is distinct from the input buffer.

Chapter 52. PyTorch Release 17.04

The NVIDIA container image of PyTorch, release 17.04, is available.

PyTorch container image version 17.04 is based on PyTorch 0.1.10.

Contents of PyTorch

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed into the `/usr/local/[bin,share,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.20
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This PyTorch release includes the following key features and enhancements.

- Reduce DataParallel overhead on more than 4 GPUs
- cuDNN v6 integration
- Synced to upstream PyTorch version as of March 2017
- Ubuntu 16.04 with March 2017 updates

Known Issues

There are no known issues in this release.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com