<table>
<thead>
<tr>
<th>Chapter</th>
<th>PyTorch Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overview</td>
</tr>
<tr>
<td>2</td>
<td>Pulling A Container</td>
</tr>
<tr>
<td>3</td>
<td>Running PyTorch</td>
</tr>
<tr>
<td>4</td>
<td>Release 20.03</td>
</tr>
<tr>
<td>5</td>
<td>Release 20.02</td>
</tr>
<tr>
<td>6</td>
<td>Release 20.01</td>
</tr>
<tr>
<td>7</td>
<td>Release 19.12</td>
</tr>
<tr>
<td>8</td>
<td>Release 19.11</td>
</tr>
<tr>
<td>9</td>
<td>Release 19.10</td>
</tr>
<tr>
<td>10</td>
<td>Release 19.09</td>
</tr>
<tr>
<td>11</td>
<td>Release 19.08</td>
</tr>
<tr>
<td>12</td>
<td>Release 19.07</td>
</tr>
<tr>
<td>13</td>
<td>Release 19.06</td>
</tr>
<tr>
<td>14</td>
<td>Release 19.05</td>
</tr>
<tr>
<td>15</td>
<td>Release 19.04</td>
</tr>
<tr>
<td>16</td>
<td>Release 19.03</td>
</tr>
<tr>
<td>17</td>
<td>Release 19.02</td>
</tr>
<tr>
<td>18</td>
<td>Release 19.01</td>
</tr>
<tr>
<td>19</td>
<td>Release 18.12</td>
</tr>
<tr>
<td>20</td>
<td>Release 18.11</td>
</tr>
<tr>
<td>21</td>
<td>Release 18.10</td>
</tr>
<tr>
<td>22</td>
<td>Release 18.09</td>
</tr>
<tr>
<td>23</td>
<td>Release 18.08</td>
</tr>
<tr>
<td>24</td>
<td>Release 18.07</td>
</tr>
<tr>
<td>25</td>
<td>Release 18.06</td>
</tr>
<tr>
<td>26</td>
<td>Release 18.05</td>
</tr>
<tr>
<td>27</td>
<td>Release 18.04</td>
</tr>
<tr>
<td>28</td>
<td>Release 18.03</td>
</tr>
<tr>
<td>29</td>
<td>Release 18.02</td>
</tr>
<tr>
<td>30</td>
<td>Release 18.01</td>
</tr>
<tr>
<td>31</td>
<td>Release 17.12</td>
</tr>
<tr>
<td>32</td>
<td>Release 17.11</td>
</tr>
<tr>
<td>33</td>
<td>Release 17.10</td>
</tr>
<tr>
<td>34</td>
<td>Release 17.09</td>
</tr>
<tr>
<td>35</td>
<td>Release 17.08</td>
</tr>
<tr>
<td>36</td>
<td>Release 17.07</td>
</tr>
<tr>
<td>37</td>
<td>Release 17.06</td>
</tr>
<tr>
<td>38</td>
<td>Release 17.05</td>
</tr>
<tr>
<td>39</td>
<td>Release 17.04</td>
</tr>
</tbody>
</table>
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

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.
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.

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.

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.

   ```bash
docker pull nvcr.io/nvidia/pytorch:20.03-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:

```bash
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:

```bash
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:

```bash
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:

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

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`
- `--shm-size=<requested memory size>`

in the command line to

```bash
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 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 2.5.6 (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 0.19.0
- MAGMA 2.5.2
- Tensor Core optimized examples:
  - ResNeXt101-32x4d
  - SE-ResNext
  - TransformerXL
  - Jasper
  - BERT
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.

<table>
<thead>
<tr>
<th>Container Version</th>
<th>Ubuntu</th>
<th>CUDA Toolkit</th>
<th>PyTorch</th>
<th>TensorRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.03</td>
<td>18.04</td>
<td>NVIDIA CUDA 10.2.89</td>
<td>1.5.0a0+8f84ded</td>
<td>TensorRT 7.0.0</td>
</tr>
<tr>
<td>20.02</td>
<td>16.04</td>
<td></td>
<td>1.5.0a0+3bb36e</td>
<td></td>
</tr>
<tr>
<td>20.01</td>
<td>16.04</td>
<td></td>
<td>1.4.0a0+a5b4d78</td>
<td></td>
</tr>
<tr>
<td>19.12</td>
<td></td>
<td>NVIDIA CUDA 10.1.243</td>
<td>1.4.0a0+174e1ba</td>
<td>TensorRT 6.0.1</td>
</tr>
<tr>
<td>19.11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.10</td>
<td></td>
<td></td>
<td>1.3.0a0+24ae9b5</td>
<td></td>
</tr>
<tr>
<td>19.09</td>
<td></td>
<td></td>
<td>1.2.0</td>
<td></td>
</tr>
<tr>
<td>19.08</td>
<td></td>
<td></td>
<td>1.2.0a0 including upstream commits up through commit 9130ab38 from July 31, 2019 as well as a cherry-picked</td>
<td>TensorRT 5.1.5</td>
</tr>
</tbody>
</table>

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. 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 the BERT: 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:

```python
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

```python
torch.logdet(W.unsqueeze(0).float()).squeeze()
```

as is done in this release.
Chapter 5.
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
PyTorch Release 20.02

- 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.

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

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](https://pytorch.org/docs/stable/amp.html).

**Tensor Core Examples**

The [tensor core examples provided in GitHub](https://github.com/NVIDIA/Tensor-Core-Examples) and [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest [deep learning example](https://github.com/NVIDIA/Tensor-Core-Examples) networks and [model scripts](https://github.com/NVIDIA/Tensor-Core-Examples) 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](https://arxiv.org/abs/1611.05431) 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](https://github.com/NVIDIA/Tensor-Core-Examples).

- **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](https://arxiv.org/abs/1709.01507) paper. This model script is available on [GitHub](https://github.com/NVIDIA/Tensor-Core-Examples).

- **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](https://github.com/magichan0718/transformer-xl) 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. This model script is available on [GitHub](https://github.com/NVIDIA/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](https://www.cs.toronto.edu/~graves/papers/2017_jasper.pdf). 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](https://github.com/NVIDIA/Tensor-Core-Examples) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/).

- **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 the [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://arxiv.org/abs/1810.04805) paper. NVIDIA’s implementation of BERT is an optimized version of the [Hugging Face implementation](https://github.com/huggingface/transformers), 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](https://github.com/NVIDIA/Tensor-Core-Examples) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/).
- **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](https://arxiv.org/abs/1707.06735), 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](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https:// github.com/NVIDIA/PyTorch).  


- **SSD300 v1.1 model.** The SSD300 v1.1 model is based on the [SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325) 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](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https://github.com/NVIDIA/PyTorch).  

- **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](https://arxiv.org/abs/1708.05557) paper. This model script is available on [GitHub](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https://github.com/NVIDIA/PyTorch).  

- **ResNet50 v1.5 model.** The ResNet50 v1.5 model is a modified version of the original [ResNet50 v1 model](https://arxiv.org/abs/1512.03385). This model script is available on [GitHub](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https://github.com/NVIDIA/PyTorch).  

- **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](https://arxiv.org/abs/1609.08144) paper. This model script is available on [GitHub](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https://github.com/NVIDIA/PyTorch).  

**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:

```python
import torch
import torch._C._jit_set_profiling_executor
import torch._C._jit_set_profiling_mode

# Disable profiling executor
torch._C._jit_set_profiling_executor(False)
# Disable profiling mode
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

```python
torch.logdet(W.unsqueeze(0).float()).squeeze()
```

as is done in this release.
Chapter 6. 
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
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.

<table>
<thead>
<tr>
<th>Container Version</th>
<th>Ubuntu</th>
<th>CUDA Toolkit</th>
<th>PyTorch</th>
<th>TensorRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.01</td>
<td>18.04</td>
<td>NVIDIA CUDA 10.2.89</td>
<td>1.4.0a0+a5b4d78</td>
<td>TensorRT 7.0.0</td>
</tr>
<tr>
<td>19.12</td>
<td>16.04</td>
<td>NVIDIA CUDA 10.1.125</td>
<td>1.4.0a0+174e1ba</td>
<td>TensorRT 6.0.1</td>
</tr>
<tr>
<td>19.11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.10</td>
<td></td>
<td>NVIDIA CUDA 10.1.243</td>
<td>1.3.0a0-24ae9b5</td>
<td></td>
</tr>
<tr>
<td>19.09</td>
<td></td>
<td></td>
<td>1.2.0</td>
<td></td>
</tr>
<tr>
<td>19.08</td>
<td></td>
<td></td>
<td>1.2.0a0 including upstream commits up through commit 9130ab3 from July 31, 2019 as well as a cherry-picked</td>
<td>TensorRT 5.1.5</td>
</tr>
</tbody>
</table>

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 the BERT: 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:

  ```python
  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

  ```python
  torch.logdet(W.unsqueeze(0).float()).squeeze()
  ```

- 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 7.  
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
- TensorFlow 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. This model script is available on [GitHub](https://github.com).

- **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](https://github.com) as well as [NVIDIA GPU Cloud (NGC)](https://docs.nvidia.com/deeplearning/gpu-cloud/).  

- **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 the [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://arxiv.org/abs/1810.04805) 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](https://github.com) as well as [NVIDIA GPU Cloud (NGC)](https://docs.nvidia.com/deeplearning/gpu-cloud/).

- **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](https://arxiv.org/abs/1708.02985). NVIDIA’s Mask R-CNN model is an optimized version of Facebook’s [implementation](https://github.com/pytorch/vision), 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](https://github.com) as well as [NVIDIA GPU Cloud (NGC)](https://docs.nvidia.com/deeplearning/gpu-cloud/).

- **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](https://arxiv.org/abs/1705.03898) paper and a flow-based neural network model from the [WaveGlow: A Flow-based Generative Network for Speech Synthesis](https://arxiv.org/abs/1803.08979) paper. This model script is available on [GitHub](https://github.com) as well as [NVIDIA GPU Cloud (NGC)](https://docs.nvidia.com/deeplearning/gpu-cloud/).

- **SSD300 v1.1 model.** The SSD300 v1.1 model is based on the [SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325) 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](https://github.com) as well as [NVIDIA GPU Cloud (NGC)](https://docs.nvidia.com/deeplearning/gpu-cloud/).

- **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](https://arxiv.org/abs/1708.05551) paper.
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 8.
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. 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 the BERT: 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).
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).

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).

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).

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 9.
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
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 the BERT: 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](https://arxiv.org/abs/1609.08144) paper. This model script is available on [GitHub](https://github.com/pytorch/pytorch) as well as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

**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 10.
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](https://arxiv.org/abs/1901.00128). 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](https://github.com/pytorch/vision) and also as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

- **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 the [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://arxiv.org/abs/1810.04805) paper. NVIDIA’s implementation of BERT is an optimized version of the [Hugging Face implementation](https://huggingface.co/transformers), 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](https://github.com/pytorch/vision) and also as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

- **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](https://arxiv.org/abs/1708.02985). NVIDIA’s Mask R-CNN model is an optimized version of [Facebook’s implementation](https://github.com/facebookresearch/maskrcnn-benchmark), 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](https://github.com/pytorch/vision) and also as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

- **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](https://arxiv.org/abs/1807.10855) paper and a flow-based neural network model from the [WaveGlow: A Flow-based Generative Network for Speech Synthesis](https://arxiv.org/abs/1807.10888) paper. This model script is available on [GitHub](https://github.com/pytorch/vision) and also as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

- **SSD300 v1.1 model.** The SSD300 v1.1 model is based on the [SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325) 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](https://github.com/pytorch/vision) and also as [NVIDIA GPU Cloud (NGC)](https://ngc.nvidia.com).

- **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...](https://arxiv.org/abs/1708.05551).
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:

```python
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`. 
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)
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](https://arxiv.org/abs/1811.00467). 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](https://github.com/pytorch/audio) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/gpu-cloud).

‣ **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 the [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://arxiv.org/abs/1810.04805) paper. NVIDIA’s implementation of BERT is an optimized version of the [Hugging Face implementation](https://huggingface.co/transformers), 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](https://github.com/pytorch/torchtext) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/gpu-cloud).

‣ **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](https://arxiv.org/abs/1708.02316). NVIDIA’s Mask R-CNN model is an optimized version of [Facebook’s implementation](https://github.com/facebookresearch/maskrcnn-benchmark), 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](https://github.com/pytorch/vision) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/gpu-cloud).


‣ **SSD300 v1.1 model.** The SSD300 v1.1 model is based on the [SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325) 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](https://github.com/pytorch/vision) as well as [NVIDIA GPU Cloud (NGC)](https://developer.nvidia.com/gpu-cloud).
- **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:

```python
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 12.
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
PyTorch Release 19.07

- 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 the [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://arxiv.org/abs/1810.04805) paper. NVIDIA’s implementation of BERT is an optimized version of the [Hugging Face implementation](https://huggingface.co/transformers), 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](https://arxiv.org/abs/1708.02947). 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.


- **SSD300 v1.1 model.** The SSD300 v1.1 model is based on the [SSD: Single Shot MultiBox Detector](https://arxiv.org/abs/1512.02325) 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](https://arxiv.org/abs/1708.05567) paper.

- **Transformer model.** The Transformer model is based on the optimized implementation in Facebook’s [Fairseq NLP Toolkit](https://github.com/pytorch/fairseq) 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](https://arxiv.org/abs/1512.03385) 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:

```python
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)
```
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
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 1.1.0 commit 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 the BERT: 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:

  ```bash
  /opt/conda/bin/conda install openblas!=0.3.6
  ```
Chapter 14.  
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
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.
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 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 16.
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
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.

- JupyterLab Server 0.2.0
- 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

- 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 17.
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
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 18.
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.
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 (master 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 20.
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 (master 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 21.
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 (master 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.
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 (master 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 23.
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.

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 fusible.
- 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 24.
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 25.
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 master 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 26.
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 master 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 27. 
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 28.
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 master 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.
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 master 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 30.
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 31. 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 32.
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 33. 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 34.
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 35.  
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 36.
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.
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 38.
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 39.
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

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS GUIDE IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the product described in this guide shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do 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 guide. NVIDIA does not accept any 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 guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NvCaffe, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, TensorRT Inference Server, Triton Inference Server, Tesla, TF-TRT, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2020 NVIDIA Corporation. All rights reserved.

www.nvidia.com