PyG

Release Notes

RN-08516-001_v24.03 | April 2024

Table of Contents

Chapter 1. PyG Overview

PyG (PyTorch Geometric) is a library built upon PyTorch to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.

It consists of various methods for deep learning on graphs and other irregular structures, also known as geometric deep learning, from a variety of published papers. In addition, it consists of easy-to-use mini-batch loaders for operating on many small and single giant graphs, multi GPU-support, DataPipe support, distributed graph learning via Quiver, a large number of common benchmark datasets (based on simple interfaces to create your own), the GraphGym experiment manager, and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds.

Chapter 2. Pulling A Container

About this task

Before you can pull a container from the NGC container registry:

- Install Docker.
  - For NVIDIA DGX™ users, see Preparing to use NVIDIA Containers Getting Started Guide.
  - For non-DGX users, see NVIDIA® GPU Cloud™ (NGC) container registry installation documentation based on your platform.
- Ensure that you have access and can log in to the NGC container registry.

  Refer to NGC Getting Started Guide for more information.

The deep learning frameworks, the NGC Docker containers, and the deep learning framework containers are stored in the `nvcr.io/nvidia` repository.

Chapter 3. Running PyG

Before you begin

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

About this task

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

- 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 by using the `NVIDIA_VISIBLE_DEVICES` environment variable).

  For more information, refer to the nvidia-docker documentation.

> **Note:** Starting in Docker 19.03, complete the steps below.

The method implemented in your system depends on the DGX OS version that you installed (for DGX systems), the NGC Cloud Image that was provided by a Cloud Service Provider, or the software that you installed to prepare to run NGC containers on TITAN PCs, Quadro PCs, or NVIDIA Virtual GPUs (vGPUs).

Procedure

1. Issue the command for the applicable release of the container that you want.

   The following command assumes that you want to pull the latest container.

   ```
   docker pull nvcr.io/nvidia/pyg:24.01-py3
   ```
2. Open a command prompt and paste the `pull` command.

Ensure that the pull successfully completes before you proceed to step 3.

3. To run the container image, select one of the following modes:

    - Interactive

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

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

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

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

    - Non-interactive

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

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

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

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

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

    To pull data and model descriptions from locations outside the container for use by PyG or save results to locations outside the container, mount one or more host directories as Docker® data volumes.

Chapter 4. PyG Release 24.03

This PyG container release is intended for use on the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 8 libraries.

Driver Requirements

Release 24.03 is based on NVIDIA CUDA 12.4.0.41, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Contents of the PyG container

This container image includes the complete source of the NVIDIA version of PyG in `/opt/pyg/pytorch_geometric`. It is prebuilt and installed as a system Python module. The /workspace/examples folder is copied from /opt/pyg/pytorch_geometric/examples for users starting to run PyG. For example, to run the gcn.py example:

```
/workspace/examples# python gcn.py
```

See `/workspace/README.md` for details.

The container also includes the following:

- torch-geometric 2.5.0
- pyg-lib 0.4.0
- This container also contains the GNN Platform (/opt/pyg/gnn-platform), an NVIDIA project that provides a low-code API for rapid GNN experimentation and training/deploying end-to-end GNN pipelines. Examples can be found at /workspace/gnn-platform-examples. For more details about the GNN Platform, see /opt/pyg/gnn-platform/README.md
- Built on PyTorch 24.03, which contains the following:
  - Ubuntu 22.04 including Python 3.10

- NVIDIA CUDA 12.4.0.41
- NVIDIA cuBLAS 12.4.2.65
- NVIDIA cuDNN 9.0.0.306
- NVIDIA NCCL 2.20
- NVIDIA RAPIDS™ 24.02
- Apex
- rdma-core 39.0
- NVIDIA HPC-X 2.18
- OpenMPI 4.1.4+
- GDRCopy 2.3
- TensorBoard 2.9.0
- Nsight Compute 2024.1.0.13
- Nsight Systems 2024.2.1.38
- NVIDIA TensorRT™ 8.6.3
- Torch-TensorRT 2.3.0a0
- NVIDIA DALI® 1.35.
- MAGMA 2.6.2
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- TransformerEngine v1.4
- PyTorch quantization wheel 2.1.2

GPU Requirements

Release 24.03 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

- New native PyG recsys example with python3 /workspace/examples/hetero/recommender_system.py.
- Includes RelBench (https://relbench.stanford.edu/) with examples for node and link level tasks. See /workspace/README.md for details.
- If trying to use /workspace/examples/ogbn_papers100m.py or /workspace/examples/multi_gpu/papers100m_gcn.py, we recommend using the respective python code from https://github.com/pyg-team/pytorch_geometric/pull/8173 instead. Please copy and paste the desired code example and run it with python3 <example_name>.py

Announcements

There are no announces for PyG in this release.

NVIDIA PyG Container Versions

The PyG container supports the same version of Ubuntu and CUDA as the PyTorch container.

Container Version	Ubuntu	CUDA Toolkit	PyG	PyTorch
24.03	22.04	NVIDIA CUDA 12.4.0.41	2.5.0	2.3.0a0+40ec155e58
24.01		NVIDIA CUDA 12.3.2	2.4.0	2.2.0a0+81ea7a4
23.11		NVIDIA CUDA 12.3.0	2.4.0	23.11
23.01	20.04	NVIDIA CUDA 12.0.1	2.2.0	23.01

Known Issues

- None

Chapter 5. PyG Release 24.02

There is no PyG container in the DLFW 24.02 release.

Chapter 6. PyG Release 24.01

This PyG container release is intended for use on the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 8 libraries.

Driver Requirements

Release 24.01 is based on CUDA 12.2.2, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Contents of the PyG container

This container image includes the complete source of the NVIDIA version of PyG in `/opt/pyg/pytorch_geometric`. It is prebuilt and installed as a system Python module. The /workspace/examples folder is copied from /opt/pyg/pytorch_geometric/examples for users starting to run PyG. For example, to run the gcn.py example:

```
/workspace/examples# python gcn.py
```

See `/workspace/README.md` for details.

The container also includes the following:

- torch-geometric 2.4.0
- pyg-lib 0.2.0
- This container also contains the GNN Platform (/opt/pyg/gnn-platform), an NVIDIA project that provides a low-code API for rapid GNN experimentation and training/deploying end-to-end GNN pipelines. Examples can be found at /workspace/gnn-platform-examples. For more details about the GNN Platform, see /opt/pyg/gnn-platform/README.md.
- Built on PyTorch 24.01, which contains the following:
    - Ubuntu 22.04 including Python 3.10

- NVIDIA CUDA® 12.3.0
- NVIDIA cuBLAS 12.3.4.1
- NVIDIA cuDNN 8.9.7.29
- NVIDIA NCCL 2.19.4
- NVIDIA RAPIDS™ 23.12
- Apex
- rdma-core 39.0
- NVIDIA HPC-X 2.16rc4
- OpenMPI 4.1.4+
- GDRCopy 2.3
- TensorBoard 2.9.0
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1.97
- NVIDIA TensorRT™ 8.6.1.6
- Torch-TensorRT 2.2.0.dev0
- NVIDIA DALI® 1.33.
- MAGMA 2.6.2
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- TransformerEngine v1.2.1
- PyTorch quantization wheel 2.1.2

GPU Requirements

Release 24.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyG release includes bug fixes, but no new features.

Announcements

There are no announces for PyG in this release.

NVIDIA PyG Container Versions

The following table shows what versions of Ubuntu, CUDA, PyG (PyTorch Geometric), and PyTorch are supported in each of the NVIDIA containers for PyG.

Container Version	Ubuntu	CUDA Toolkit	PyG	PyTorch
24.01	22.04	NVIDIA CUDA 12.3.2	2.4.0	2.2.0a0+81ea7a4
23.11		NVIDIA CUDA 12.3.0	2.4.0	23.11
23.01	20.04	NVIDIA CUDA 12.0.1	2.2.0	23.01

Known Issues

- torch.compile currently supports x86 systems only.

Chapter 7. PyG Release 23.12

There is no PyG container in DLFW release 23.12.

Chapter 8. PyG Release 23.11

This PyG container release is intended for use on the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 8 libraries.

Driver Requirements

Release 23.11 is based on CUDA 12.2.2, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Contents of the PyG container

This container image includes the complete source of the NVIDIA version of PyG in `/opt/pyg/pytorch_geometric`. It is prebuilt and installed as a system Python module. The /workspace/examples folder is copied from /opt/pyg/pytorch_geometric/examples for users starting to run PyG. For example, to run the gcn.py example:

```
/workspace/examples# python gcn.py
```

See `/workspace/README.md` for details.

The container also includes the following:

- torch-geometric 2.4.0
- pyg-lib 0.2.0
- This container also contains the GNN Platform (/opt/pyg/gnn-platform), an NVIDIA project that provides a low-code API for rapid GNN experimentation and training/deploying end-to-end GNN pipelines. Examples can be found at /workspace/gnn-platform-examples. For more details about the GNN Platform, see /opt/pyg/gnn-platform/README.md.
- Built on PyTorch 23.11, which contains the following:
    - Ubuntu 22.04 including Python 3.10

- NVIDIA CUDA® 12.3.0
- NVIDIA cuBLAS 12.3.2.1
- NVIDIA cuDNN 8.9.6.50
- NVIDIA NCCL 2.19.3
- NVIDIA RAPIDS™ 23.10
- Apex
- rdma-core 39.0
- NVIDIA HPC-X 2.16
- OpenMPI 4.1.4+
- GDRCopy 2.3
- TensorBoard 2.9.0
- Nsight Compute 2023.3.0.12
- Nsight Systems 2023.3.1.92
- NVIDIA TensorRT™ 8.6.1.6
- Torch-TensorRT 2.2.0.dev0
- NVIDIA DALI® 1.31.0
- MAGMA 2.6.2
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- TransformerEngine 1.0
- PyTorch quantization wheel 2.2.0

GPU Requirements

Release 23.11 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyG release includes the following key features and enhancements.

- Torch-frame integration.
- torch.compile accelerations: We recommend using torch.compile on your GNN models for accelerating any example.
  - Example: '`model=torch.compile(model)`'
- NVIDIAs syngen tool for synthetic graph data generation. See README.md for details.

Announcements

General availability starts from 23.11.

NVIDIA PyG Container Versions

The following table shows what versions of Ubuntu, CUDA, PyG (PyTorch Geometric), and PyTorch are supported in each of the NVIDIA containers for PyG.

Container Version	Ubuntu	CUDA Toolkit	PyG	PyTorch
23.11	22.04	NVIDIA CUDA 12.3.0	2.4.0	23.11
23.01	20.04	NVIDIA CUDA 12.0.1	2.2.0	23.01

Examples

There is an extensive suite of examples provided by PyG stored at `/workspace/examples`.

To start, the most basic example is `gcn.py`. For examples on basic multi-GPU and multi-node usage, see `multi_gpu/distributed_sampling.py` and `multi_gpu/distributed_sampling_multinode.py`. For guidance on scaling up to larger data try `ogbn_papers_100m.py` from the examples folder. To scale this up to use all of the GPUs on a single node, try `multi_gpu/singlenode_multigpu_papers100m_gcn.py`. To scale further to multi-node, run `multi_gpu/multinode_multigpu_papers100m_gcn.py` using the slurm commands at the top of the file.

Additionally, NVIDIA has created a GNN Platform which consists of a high level API for training and deploying end to end GNN workflows. Detailed ipython notebook examples can be found at `/workspace/gnn-platform-examples`. Additional example GNN Platform workflows can be found at `/opt/pyg/gnn-platform/tests`.

In order to work with ipython notebooks make sure to launch your docker containers with the --network=host --ipc=host flags in your docker run command. For more details on working with the gnn-platform see README at `/opt/pyg/gnn-platform/README.md`.

Known Issues

- On Arm-based systems, PyG's `gdc` function encounters numerical errors when tested with `test_gdc`. Open Issue: #7431.

.

Chapter 9. PyG Release 23.01

This PyG container release is intended for use on the NVIDIA® Ampere Architecture GPU, NVIDIA A100, and the associated NVIDIA CUDA® 12 and NVIDIA cuDNN 8 libraries.

Driver Requirements

Release 23.01 is based on CUDA 12.0.1, which requires NVIDIA Driver release 525 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), or 525.85 (or later R525).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 12.0. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Contents of the PyG container

This container image includes the complete source of the NVIDIA version of DGL in `/opt/pyg/pytorch_geometric`. It is prebuilt and installed as a system Python module

The container also includes the following:

- torch-geometric 2.2.0
- RAPIDS 22.12
- NVIDIA CUDA® 12.0.1
- NVIDIA cuBLAS from CUDA 12.0.1
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.16.5 (optimized for NVIDIA NVLink®)
- Apex
- rdma-core 36.0
- NVIDIA HPC-X 2.13
- OpenMPI 4.1.4+
- GDRCopy 2.3
- TensorBoard 2.9.0

- Nsight Compute 2022.4.1.6
- Nsight Systems 2022.5.1
- NVIDIA TensorRT™ 8.5.2.2
- Torch-TensorRT 1.4.0dev0
- NVIDIA DALI® 1.21.0
- MAGMA 2.6.2
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- PyTorch quantization wheel 2.1.2
- TransformerEngine 0.4

GPU Requirements

Release 23.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This PyG release includes the following key features and enhancements.

- PyG container image version 23.01 is based on PyTorch Geometric 2.2.0. The major features of the release can be found in the PyG release notes.

NVIDIA PyG Container Versions

The following table shows what versions of Ubuntu, CUDA, PyG (PyTorch Geometric), and PyTorch are supported in each of the NVIDIA containers for PyG.

Container Version	Ubuntu	CUDA Toolkit	PyG	PyTorch
23.01	20.04	NVIDIA CUDA 12.0.1	2.2.0	23.01

Known Issues

None.

Notice

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

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

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

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

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

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

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

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

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

HDMI

OpenCL

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

Trademarks

Copyright

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