Installation

We have outsourced a lot of functionality of PyTorch Geometric to other packages, which needs to be installed in advance. These packages come with their own CPU and GPU kernel implementations based on the newly introduced C++/CUDA extensions in PyTorch 0.4.0.

> **Note**
>
> We do not recommend installation as root user on your system python. Please setup an Anaconda/Miniconda environment or create a Docker image.

Please follow the steps below for a successful installation:

1. Ensure that at least PyTorch 1.0.0 is installed:

```
$ python -c "import torch; print(torch.__version__)"
>>> 1.0.0
```

2. Ensure CUDA is setup correctly (optional):

1. Check if PyTorch is installed with CUDA support:

```
$ python -c "import torch; print(torch.cuda.is_available())"
>>> True
```

2. Add CUDA to `$PATH` and `$CPATH` (note that your actual CUDA path may vary from `/usr/local/cuda`):

```
$ PATH=/usr/local/cuda/bin:$PATH
$ echo $PATH
>>> /usr/local/cuda/bin:...

$ CPATH=/usr/local/cuda/include:$CPATH
$ echo $CPATH
>>> /usr/local/cuda/include:...
```

3. Add CUDA to `$LD_LIBRARY_PATH` on Linux and to `$DYLD_LIBRARY_PATH` on macOS (note that your actual CUDA path may vary from `/usr/local/cuda`):

```
$ LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
$ echo $LD_LIBRARY_PATH
>>> /usr/local/cuda/lib64:...

$ DYLD_LIBRARY_PATH=/usr/local/cuda/lib:$DYLD_LIBRARY_PATH
$ echo $DYLD_LIBRARY_PATH
>>> /usr/local/cuda/lib:...
```

4. Verify that `nvcc` is accessible from terminal:

```
$ nvcc --version
>>> 10.0
```

5. Ensure that PyTorch and system CUDA versions match:

```
$ python -c "import torch; print(torch.version.cuda)"
>>> 10.0

$ nvcc --version
>>> 10.0
```

3. Install all needed packages:

```
$ pip install --verbose --no-cache-dir torch-scatter
$ pip install --verbose --no-cache-dir torch-sparse
$ pip install --verbose --no-cache-dir torch-cluster
$ pip install --verbose --no-cache-dir torch-spline-conv (optional)
$ pip install torch-geometric
```

In rare cases, CUDA or Python path problems can prevent a successful installation. `pip` may even signal a successful installation, but runtime errors complain about missing modules, *.e.g.*, `No module named 'torch_*.*_cuda'`, or execution simply crashes with `Segmentation fault (core dumped)`. We collected a lot of common installation errors in the Frequently Asked Questions subsection. In case the FAQ does not help you in solving your problem, please create an issue. You should additionally verify that your CUDA is set up correctly by following the official installation guide, and that the official extension example runs on your machine.

C++/CUDA Extensions on macOS

In order to compile CUDA extensions on macOS, you need to replace the call

```
def spawn(self, cmd):
    spawn(cmd, dry_run=self.dry_run)
```

with

```
import subprocess

def spawn(self, cmd):
    subprocess.call(cmd)
```

in `lib/python{xxx}/distutils/ccompiler.py`.

Frequently Asked Questions

1. `ImportError: ***: cannot open shared object file: No such file or directory`: Add CUDA to your `$LD_LIBRARY_PATH` (see Issue#43).
2. `undefined symbol:`, *e.g.* `_ZN2at6detail20DynamicCUDAInterface10set_deviceE`: Clear the pip cache and reinstall the respective package (see Issue#7). On macOS, it may help to install clang compilers via conda (see Issue#18):

   ```
   $ conda install -y clang_osx-64 clangxx_osx-64 gfortran_osx-64
   ```

3. Unable to import `*_cuda`: You need to `import torch` first before importing any of the extension packages (see Issue#6).
4. `error: command '/usr/bin/nvcc' failed with exit status 2`: Ensure that at least CUDA >= 8 is installed (see Issue#25a and Issue#106).
5. `return __and_<is_constructible<_Elements, _UElements&&>...>::value`: Ensure that your `gcc` version is at least 4.9 (and below 6) (see Issue#25b). You will also need to reinstall PyTorch because `gcc` versions must be consistent across all PyTorch packages.
6. `file not recognized: file format not recognized`: Clean the repository and temporarily rename Anaconda's `ld` linker (see Issue#16683).
7. `undefined symbol: __cudaPopCallConfiguration`: Ensure that your PyTorch CUDA version and system CUDA version match (see Issue#19):

```
$ python -c "import torch; print(torch.version.cuda)"
$ nvcc --version
```

8. `undefined symbol: _ZN3c105ErrorC1ENS_14SourceLocationERKSs`: The `std::string` abi does not match between building PyTorch and its extensions. This is fixable by building extensions with `-D_GLIBCXX_USE_CXX11_ABI=1` or building PyTorch from source (see this PyTorch thread).
9. On macOS: `'gcc' failed with exit status 1`: Install the respective packages by using the following environment variables (see Issue#21):

```
$ MACOSX_DEPLOYMENT_TARGET=10.9 CC=clang CXX=clang++ python setup.py install
```

10. On macOS: `ld: warning: directory not found for option '-L/usr/local/cuda/lib64'` and `ld: library not found for -lcudart`: Symlink `cuda/lib` to `cuda/lib64` (see Issue#116):

```
$ sudo ln -s /usr/local/cuda/lib /usr/local/cuda/lib64
```

11. On macOS: `The version of the host compiler ('Apple clang') is not supported`: Downgrade your command line tools (see this StackOverflow thread) with the respective version annotated in the CUDA Installation Guide for Mac (Section 1.1) for your specific CUDA version. You can download previous command line tool versions here.