Windows FAQ

Building from source

Include optional components

There are two supported components for Windows PyTorch: MKL and MAGMA. Here are the steps to build with them.

```
REM Make sure you have 7z and curl installed.

REM Download MKL files
curl https://s3.amazonaws.com/ossci-windows/mkl_2020.2.254.7z -k -O
7z x -aoa mkl_2020.2.254.7z -omkl

REM Download MAGMA files
REM version available:
REM 2.5.4 (CUDA 10.1 10.2 11.0 11.1) x (Debug Release)
REM 2.5.3 (CUDA 10.1 10.2 11.0) x (Debug Release)
REM 2.5.2 (CUDA 9.2 10.0 10.1 10.2) x (Debug Release)
REM 2.5.1 (CUDA 9.2 10.0 10.1 10.2) x (Debug Release)
set CUDA_PREFIX=cuda102
set CONFIG=release
curl -k https://s3.amazonaws.com/ossci-windows/magma_2.5.4_%CUDA_PREFIX%_%CONFIG%.7z -o magma.7z
7z x -aoa magma.7z -omagma

REM Setting essential environment variables
set "CMAKE_INCLUDE_PATH=%cd%\mkl\include"
set "LIB=%cd%\mkl\lib;%LIB%"
set "MAGMA_HOME=%cd%\magma"
```

Speeding CUDA build for Windows

Visual Studio doesn't support parallel custom task currently. As an alternative, we can use `Ninja` to parallelize CUDA build tasks. It can be used by typing only a few lines of code.

```
REM Let's install ninja first.
pip install ninja

REM Set it as the cmake generator
set CMAKE_GENERATOR=Ninja
```

One key install script

You can take a look at this set of scripts. It will lead the way for you.

Extension

CFFI Extension

The support for CFFI Extension is very experimental. You must specify additional `libraries` in `Extension` object to make it build on Windows.

```
ffi = create_extension(
    '_ext.my_lib',
    headers=headers,
    sources=sources,
    define_macros=defines,
    relative_to=__file__,
    with_cuda=with_cuda,
    extra_compile_args=["-std=c99"],
    libraries=['ATen', '_C'] # Append cuda libraries when necessary, like cudart
)
```

Cpp Extension

This type of extension has better support compared with the previous one. However, it still needs some manual configuration. First, you should open the **x86_x64 Cross Tools Command Prompt for VS 2017**. And then, you can start your compiling process.

Installation

Package not found in win-32 channel.

```
Solving environment: failed

PackagesNotFoundError: The following packages are not available from current channels:

- pytorch

Current channels:
- https://conda.anaconda.org/pytorch/win-32
- https://conda.anaconda.org/pytorch/noarch
- https://repo.continuum.io/pkgs/main/win-32
- https://repo.continuum.io/pkgs/main/noarch
- https://repo.continuum.io/pkgs/free/win-32
- https://repo.continuum.io/pkgs/free/noarch
- https://repo.continuum.io/pkgs/r/win-32
- https://repo.continuum.io/pkgs/r/noarch
- https://repo.continuum.io/pkgs/pro/win-32
- https://repo.continuum.io/pkgs/pro/noarch
- https://repo.continuum.io/pkgs/msys2/win-32
- https://repo.continuum.io/pkgs/msys2/noarch
```

PyTorch doesn't work on 32-bit system. Please use Windows and Python 64-bit version.

Import error

```
from torch._C import *

ImportError: DLL load failed: The specified module could not be found.
```

The problem is caused by the missing of the essential files. Actually, we include almost all the essential files that PyTorch need for the conda package except VC2017 redistributable and some mkl libraries. You can resolve this by typing the following command.

```
conda install -c peterjc123 vc vs2017_runtime
conda install mkl_fft intel_openmp numpy mkl
```

As for the wheels package, since we didn't pack some libraries and VS2017 redistributable files in, please make sure you install them manually. The VS 2017 redistributable installer can be downloaded. And you should also pay attention to your installation of Numpy. Make sure it uses MKL instead of OpenBLAS. You may type in the following command.

```
pip install numpy mkl intel-openmp mkl_fft
```

Another possible cause may be you are using GPU version without NVIDIA graphics cards. Please replace your GPU package with the CPU one.

```
from torch._C import *

ImportError: DLL load failed: The operating system cannot run %1.
```

This is actually an upstream issue of Anaconda. When you initialize your environment with conda-forge channel, this issue will emerge. You may fix the intel-openmp libraries through this command.

```
conda install -c defaults intel-openmp -f
```

Usage (multiprocessing)

Multiprocessing error without if-clause protection

```
RuntimeError:
       An attempt has been made to start a new process before the
       current process has finished its bootstrapping phase.

   This probably means that you are not using fork to start your
   child processes and you have forgotten to use the proper idiom
   in the main module:

       if __name__ == '__main__':
           freeze_support()
           ...

   The "freeze_support()" line can be omitted if the program
   is not going to be frozen to produce an executable.
```

The implementation of `multiprocessing` is different on Windows, which uses `spawn` instead of `fork`. So we have to wrap the code with an if-clause to protect the code from executing multiple times. Refactor your code into the following structure.

```
import torch

def main()
    for i, data in enumerate(dataloader):
        # do something here

if __name__ == '__main__':
    main()
```

Multiprocessing error "Broken pipe"

```
ForkingPickler(file, protocol).dump(obj)

BrokenPipeError: [Errno 32] Broken pipe
```

This issue happens when the child process ends before the parent process finishes sending data. There may be something wrong with your code. You can debug your code by reducing the `num_worker` of :class:`~torch.utils.data.DataLoader` to zero and see if the issue persists.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\notes\(pytorch-master)(docs)(source)(notes)windows.rst`, line 223); *backlink*

Unknown interpreted text role "class".

Multiprocessing error "driver shut down"

```
Couldnâ€™t open shared file mapping: <torch_14808_1591070686>, error code: <1455> at torch\lib\TH\THAllocator.c:154

[windows] driver shut down
```

Please update your graphics driver. If this persists, this may be that your graphics card is too old or the calculation is too heavy for your card. Please update the TDR settings according to this post.

CUDA IPC operations

```
THCudaCheck FAIL file=torch\csrc\generic\StorageSharing.cpp line=252 error=63 : OS call failed or operation not suppo
```

They are not supported on Windows. Something like doing multiprocessing on CUDA tensors cannot succeed, there are two alternatives for this.

1. Don't use `multiprocessing`. Set the `num_worker` of :class:`~torch.utils.data.DataLoader` to zero.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\notes\(pytorch-master)(docs)(source)(notes)windows.rst`, line 252); *backlink*

Unknown interpreted text role "class".

2. Share CPU tensors instead. Make sure your custom :class:`~torch.utils.data.DataSet` returns CPU tensors.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\pytorch-master\docs\source\notes\(pytorch-master)(docs)(source)(notes)windows.rst`, line 255); *backlink*

Unknown interpreted text role "class".