Enable GPU support with Pytorch (macOS)

This tutorial is to enable the use of the GPU in the Macbooks available on the lockers. All of these computers have Python and Anaconda already installed, so if you are using your personal computer, make sure that you have that installed before starting this tutorial. Please take a look at the other wiki pages about installing both Python and Anaconda.

Requirements:

- Macbook with M1/M2/M3 chip available
- macOS 12.3 or later
- Python 3.8 or later
- Package manager such as Anaconda or pip.

Steps:

1. Open your terminal and create a new virtual environment with:
   `conda create -n ENV_NAME python=PYTHON_VERSION -y`

Replace **ENV_NAME** with the name of your environment. The name can be anything, just make sure that it is not being used already. You can check that by typing `conda env list`. Also, remember that the name has to be a single compound word, and **no spaces**, for example, `my_environment`, or `pytorch_env` are good. `my environment` and `pytorch env` are not.

For the Python version, check for the best version for all the other libraries you need for your project.

2. Activate your environment from your terminal with:
   `conda activate ENV_NAME`
3. Install PyTorch by running:
   `pip3 install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/nightly/cpu`

This will install Pytorch and all the important dependencies in that library, like torchvision and torchaudio (to this, you can additionally add torchtext).

4. Test your install by running the next Python code. The output of it should be `tensor([1.], device='mps:0')`

```
import torch
if torch.backends.mps.is_available():
    mps_device = torch.device("mps")
```

```
    x = torch.ones(1, device=mps_device)
    print (x)
else:
    print ("MPS device not found.")
```

5. **Additional information:** Pytorch has a lot of documentation of its different modules, but a very useful one is to know how to assign hardware as a backend (this means it helps you set up and run operations on the GPU). For the `mps` device, we assign it like in the next example:

```
# Check that MPS is available
if not torch.backends.mps.is_available():
    if not torch.backends.mps.is_built():
        print("MPS not available because the current PyTorch install was not "
              "built with MPS enabled.")
    else:
        print("MPS not available because the current MacOS version is not 12.3+ "
              "and/or you do not have an MPS-enabled device on this machine.")

else:
    mps_device = torch.device("mps")

    # Create a Tensor directly on the mps device
    x = torch.ones(5, device=mps_device)
    # Or
    x = torch.ones(5, device="mps")

    # Any operation happens on the GPU
    y = x * 2

    # Move your model to mps just like any other device
    model = YourFavoriteNet()
    model.to(mps_device)

    # Now every call runs on the GPU
    pred = model(x)
```

Revision #1
Created 25 March 2024 10:26:02 by Mayra Cristina Berrones Reyes
Updated 29 April 2024 22:22:38 by Mayra Cristina Berrones Reyes