Setting up Tensorflow and PyTorch with GPU support (on a variety of platforms)

(v0.1 of these instructions, current as of May 2025)

GPU support is essential for deep learning; computation that is very time consuming on a CPU (even a very fast one!) is much faster on a GPU, which is designed to do many calculations in parallel.

GPU support requires:

1. Hardware that is capable;
2. Correct setup of the operating system and development environment.

The first point above is often the key issue. Most Windows machines (especially laptops) have only 'integrated graphics' support, which essentially means that a (fairly weak) GPU is included in the CPU/chipset itself; it does not support the kinds of computation needed for efficient deep learning. Machines with suitable hardware include:

- Windows machines with reasonably powerful (and most often, NVIDIA) GPUs. Most commonly, these would be either 'gaming laptops', or desktop machines with powerful graphics cards.
    - AMD GPUs also support this type of computation, but AMD has been playing 'catch up' to NVIDIA in this area. While NVIDIA is universally supported, using AMD will likely require some work and research.
- Apple Macs with Apple Silicon processors (see the notes below).

If you don't have suitable hardware, then Google Colab is likely your best choice (and might be best, in any case).

Google Colab

Many of you are already familiar with Google Colab, a cloud-based platform allowing you to write and execute code (Jupyter notebooks) without any installation on your local machine.

While Colab provides CPU-only support by default, it can be configured to use a GPU. (It is remarkable that Google provides free access to GPU computing resources!)

Of course, as a cloud service, it has the usual drawbacks: requires you to be online, imposes certain restrictions (such as computation runtime being limited), doesn't provide direct control of

the environment, etc. (While the resources are limited at the 'free' level, you have the option of paying for a subscription, if you choose.)

However, Colab is by far the easiest platform to set up, and despite the limitations noted above, may be the best choice for this term.

To turn on GPU support:

1. Access the Colab notebook editor.
2. Near the top-right, you will see a small meter which shows RAM and disk usage. Click the arrow.
3. In the dropdown box, click 'Change runtime type.'
4. Change the hardware accelerator from 'CPU' to 'T4 GPU', and save your selection.

Windows

Many of you use Windows on your machine. If you have an adequate graphics card, it is possible to enable it for TensorFlow/PyTorch use. However, it is somewhat more challenging than with other platforms.

To install directly, you essentially must first set up WSL, which allows Linux to be run from within Windows, and then complete most of the same steps at the Linux install. I have found this to be the most challenging installation, and very error prone (there are lots of different instructions, lots of different errors as versions change, etc.)

If you want to work in Windows, I would strongly recommend that you use Docker to do so. Docker allows you to (in very simple terms) run virtual servers on your machine. Here, we set up a pre-configured Docker server with our running Python environment, including Tensorflow/PyTorch. We can then connect to, e.g., JupyterLab running on the server.

Docker

First, a bit of nomenclature.

- An *image* is similar to a snapshot of a server, already set up with software, ready to be instantiated.
- A *container* is an instance of a server, created from an image.
- A *Docker compose* file specifies the configuration of server(s), including which image(s) should be used/downloaded and how they should be set up.

Along with this document, you have been provided with a zip file containing a compose file (and other related setup files). This file specifies most of the setup required to get a working

container with GPU support up and running. I have included setup for both TensorFlow and PyTorch.

(This set of instructions, and the accompanying files, are based mostly on material from this site and especially this github site. This setup has been modified for our purposes, and is not intended to be particularly well written or robust. It may break in the future.)

1. Ensure that you have up-to-date GPU drivers installed: NVIDIA GPU drivers
2. From a Windows command prompt, run `wsl --update`, to set up the Linux platform on your Windows machine.
3. Install Docker Desktop. (Note the installation requirements on that page.)
4. Run Docker Desktop (if it doesn't start automatically).
5. Unzip the provided docker setup file in the place you wish to work. (As currently configured, this folder will contain both your configuration files and your work. You will also need to be able to navigate to it at the command line. So locate it accordingly on your machine!)
6. Open a command prompt window, and navigate to the top level of the unzipped folder (i.e., your active directory should be the same one as the docker-compose.yml file).
7. `docker-compose up` (The first time you run this command, it will take a long time, as it downloads the necessary software.)
8. Once the container is up and running, you should see a message indicating URLs that can be used to access JupyterLab. Copy one of the URLs to a web browser's address bar. (The one beginning with 127.0.0.1 is the safest choice--it should work properly regardless of your network configuration.) JupyterLab should open in the browser.

(Once things have been set up, you can monitor and control them via the Docker Desktop application, if you wish. You can look in the Docker Desktop application, and you should see both the image, and the running container.)

You can stop the container by pressing Control-C in the command window where you started the server, or by pressing the stop button beside the container in Docker Desktop.

Each time you want to start the server and JupyterLab, you will need to repeat from step 6. (Note that you can't simply bookmark JupyterLab page in your browser, because the token will change each time you start the software.)

(Rather than using the command line, you can also restart the container by running it from within Docker Desktop. In this case, you will need to 'view details' for the running container to see the text output, including the necessary URLs.)

Files that are placed within the data/notebooks subfolders of your Windows folder, will be available within Jupyterlab.

Direct install

I did a lot of testing (and writing of instructions!) for direct installs in Windows. Honestly, it was just too difficult to reliably get these working. (I believe there are issues/conflicts with the latest versions of the components, but I only have so much time to test other permutations.) I don't want to set you up for potential frustration, so I removed these instructions.

I would strongly recommend that you do a Docker install instead. If you want to pursue a direct install, you can find many guides online.

Linux

It is worth noting that, while you might not be very familiar with Linux, this operating system is the most common choice for this type of work. (Installing in Windows normally entails getting Windows' Linux environment (i.e., WSL) working; typically, it is easier to just work directly in Linux.) It is worth becoming comfortable with Linux.

If you wish to pursue this option, and you don't already have a Linux machine (with GPU) set up, then you will generally need to install a Linux distribution on your machine. (There are two primary options: you might install Linux as the only OS on a machine, or you might set it up to 'dual-boot' with Windows, so that you have a choice of OS when you boot your machine. Dual-boot is the most common choice for Windows users who want to use Linux as well.) Ubuntu is the most common Linux distribution for end-user machines.

On Linux, you have the choice of installing either directly, or using a Docker container.

Direct install

Tensorflow
(Based on the instructions here (be sure the Linux tab is selected).)

The following steps are applicable to any running Python environment, but we assume that you have a running Anaconda install, with a fresh Python environment.

1. Ensure that you have the following installed (with up-to-date versions):
    1. NVIDIA GPU drivers
    2. NVIDIA CUDA Toolkit
    3. NVIDIA cuDNN
2. Enter your Python environment (e.g., an Anaconda environment) command line.
3. `pip install tensorflow[and-cuda]`

PyTorch
(Based on instructions here.)

Continuing from the TensorFlow installation above:

```
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu128
```

(The command above is a single line, not two separate commands. It is based on the versions available at the time this was written, and may be out of date. Check the instruction link for an up-to-date version.)

Docker

If you encounter problems, Docker installs might be easier to get running. You can find instructions online (e.g., for Tensorflow, here). The instructions for Windows, above, may be useful as well.

Mac

If you have a Mac with an Apple Silicon processor (i.e., M1 and its successors), then the GPU capabilities of the processor can be employed for machine learning. The less powerful models (like the M1) will be fairly slow. However, more recent models (particularly the more powerful variants--the Pro/Max/Ultra versions) compare reasonably well to dedicated NVidia GPUs. (For comparison, the TensorFlow model in the test script took approximately 7ms per step on my M3 Pro, roughly double the time that Colab took (3ms/step).

As with other platforms, It is possible to use docker images on Mac (for example, see here). However, direct installation is simple and effective enough that I would recommend it over using Docker.

Direct Install

The following steps are applicable to any running Python environment, but we assume that you have a running Anaconda install, with a fresh Python environment.

Install Tensorflow with GPU support (based on these instructions):

```
python -m pip install tensorflow
python -m pip install tensorflow-macos
python -m pip install tensorflow-metal
```

Then verify with the provided notebook (or with the script at the link above).

Install PyTorch with GPU support (based on these instructions):

```
conda install pytorch torchvision torchaudio -c pytorch-nightly
```

Then verify with the provided notebook (or with the script at the link above).