Setting Up The Working Environment

This guide has 3 parts:

1. Setting up Anaconda environment and Python (+PyTorch) locally.
2. Setting up PyCharm to work in the Anaconda environment.
3. Setting up Google Colab with GPU.

The guide may seem long, but it is mostly images, so don't be afraid 😊

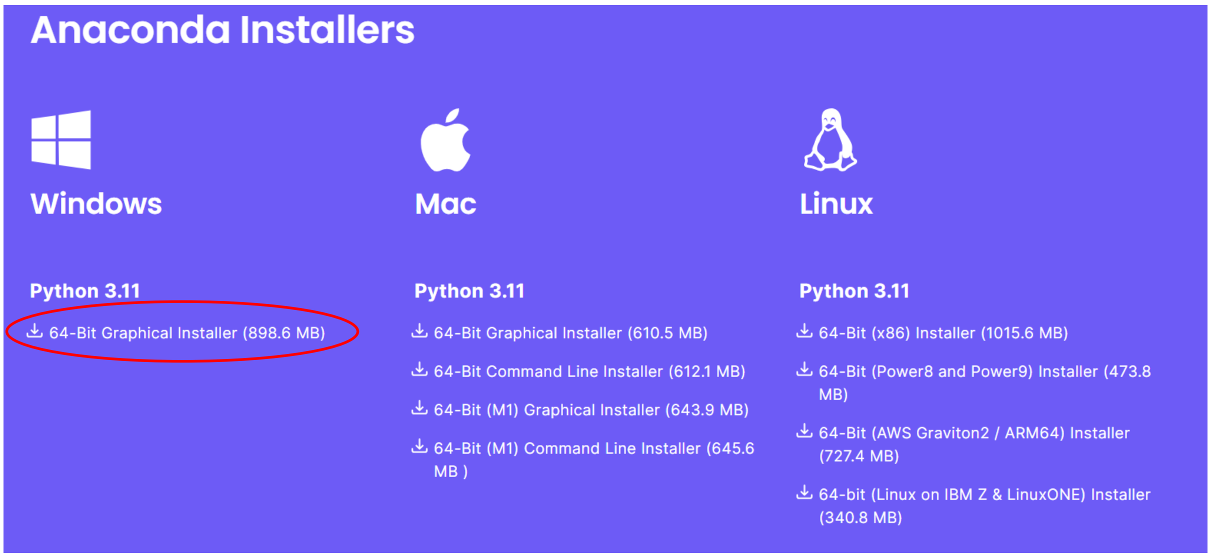
Relevant courses: 046746, 046202, 046211.

Part 1 - Setting up Anaconda environment and Python (+PyTorch) locally

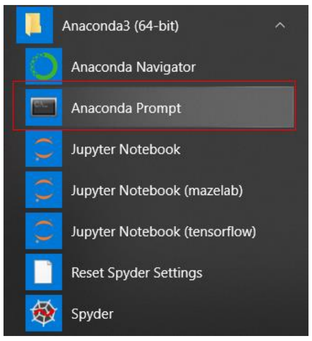
In this part, we are going to download Anaconda Navigator with Python 3, create a new virtual environment for the course and install the required libraries.

Windows (But it is the same for Linux, just download the correct installer)

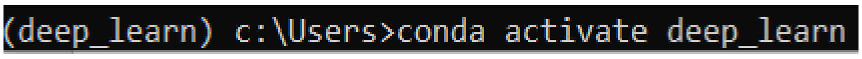
1. Go to https://www.anaconda.com/download and download the installer. Make sure to pick Windows and the Python 3 version (if you are on Linux, you can also "curl" or "wget" to download the ".sh" file).


Anaconda Installers
Windows
Mac
Linux
Python 3.11
64-Bit Graphical Installer (898.6 MB)
Python 3.11
64-Bit Graphical Installer (610.5 MB)
64-Bit Command Line Installer (612.1 MB)
64-Bit (M1) Graphical Installer (643.9 MB)
64-Bit (M1) Command Line Installer (645.6 MB)
Python 3.11
64-Bit (x86) Installer (1015.6 MB)
64-Bit (Power8 and Power9) Installer (473.8 MB)
64-Bit (AWS Graviton2 / ARM64) Installer (727.4 MB)
64-bit (Linux on IBM Z & LinuxONE) Installer (340.8 MB)

2. Launch the installer and follow the instructions on-screen. Important note: the path in which you install Anaconda **MUST NOT** include any characters other than English (if the username of your Windows is in Hebrew, please pick an English-only path).
3. Once installation is done, we need to create a new environment. Launch "Anaconda Prompt" from the start menu (Linux users: just open the Terminal).

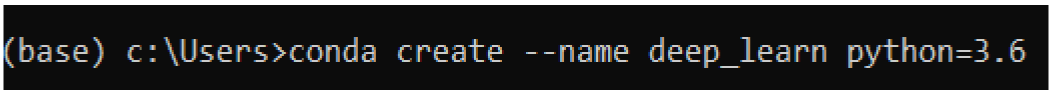


4. Creating a new environment and installing the required the packages can be done manually (steps **5-7**) or automatically using the provided `environment.yml` file.
    a. Navigate to the directory where you downloaded the `environment.yml` file and run `*conda env create -f environment.yml*`.
    b. After the process is done, you can activate the new environment by typing "*conda activate env_name*". You will see the name in the parenthesis change.



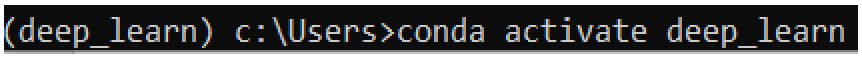
```
(deep_learn) c:\Users>conda activate deep_learn
```

    c. If you choose the automatic approach, you will still need to install PyTorch manually – jump to step **8**.
    d. To see installed packages, type `*conda list -n deep_learn*`.
    e. *General guide to managing environments (creating, removing, cloning…)*
5. In the command line you will see the name of the current environment (the basic one is called "base") on the left. Type "*conda create --name env_name python=3.8*", where "*env_name*" is the name of the environment and the Python version can be 3.7 or 3.8. Pick any name you want ("torch", "ml", "deep_learn", "anam"….) and press Enter.



```
(base) c:\Users>conda create --name deep_learn python=3.6
```

(To remove an environment*:* ` conda remove --name myenv --all`
6. After the process is done, you can activate the new environment by typing "*conda activate env_name*". You will see the name in the parenthesis change.



```
(deep_learn) c:\Users>conda activate deep_learn
```

7. Now we are going to install new libraries. Activate the new environment. All the libraries and their installation commands can be found here: https://anaconda.org/ (if in the future you want to install other libraries, start your search there. There are libraries for Computer Vision, Speech, Reinforcement Learning…). Here is the list of libraries you need to install. Run the commands in Anaconda Prompt while the new environment is activated.

Library	Command to Run
Jupyter Notebook	`conda install -c conda-forge notebook`
Numpy	`conda install -c conda-forge numpy`
Matplotlib	`conda install -c conda-forge matplotlib`
OpenCV	`conda install -c conda-forge opencv`
Scipy	`conda install -c anaconda scipy`
Scikit-Learn	`conda install -c conda-forge scikit-learn`
Pandas	`conda install -c conda-forge pandas`
Tqdm	`conda install -c conda-forge tqdm`
Plotly	`conda install -c conda-forge plotly`
Seaborn (046202)	`conda install -c conda-forge seaborn`
Optuna (046211)	`pip install optuna`

Note: `**pip**` commands also work here! For example, to install `torchviz`: `pip install torchviz`, just make sure the correct environment is activated.
If you are getting SSL errors when trying to install, check out this thread. Note that the Anaconda library is usually located (in Windows) here*: C:\ProgramData\Anaconda3*
8. Installing PyTorch – we are now going to install PyTorch. It has a CPU version and a GPU version (which also includes the CPU version). If you know that you have an Nvidia GPU that supports CUDA, you need to install the CUDA drivers. Follow this guide:
    - Windows: https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html

- Linux: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html

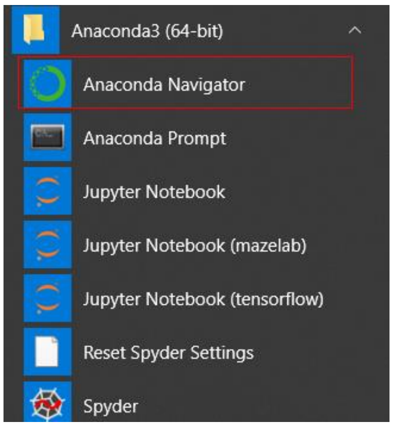
A list of supported GPUs for deep learning can be found here:
https://developer.nvidia.com/cuda-gpus
To customize your installation, you can visit PyTorch homepage as they have great UI:
https://pytorch.org/get-started/locally/

Finally, here are the commands you need to run (for GPU, validate your CUDA version!):

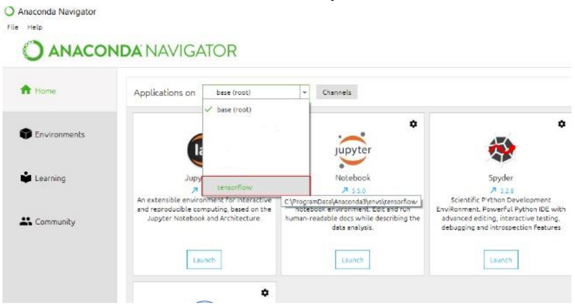
Library	Command to Run
PyTorch CPU	`conda install pytorch torchvision torchaudio cpuonly -c pytorch`
Pytorch GPU (only if you have one!)	`conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia` (Note: to check the *maximal* supported CUDA version for the installed NVIDIA drivers on your machine, open Terminal/CMD/PowerShell and run `nvidia-smi`. The maximal supported CUDA version will appear at the top-right side of the table.
TorchText (046211)	`conda install -c pytorch torchtext`

After you are done installing everything you can clean the cache by running:
`conda clean --all`

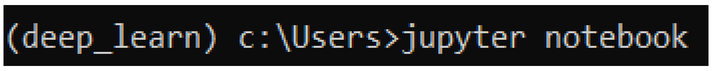
9. Great, we now have everything installed and we can launch Notebooks locally. Launch Anaconda Navigator from the start menu.



10. Switch to the environment you created from the scroll menu.



11. Now you can launch Jupyter Notebook which will open in your default browser. Now you can download the tutorials and homeworks and run them locally. You can also install other applications straight from Navigator (PyCharm, VSCode, JupyterLab). You can also run the notebook from the command line by running `jupyter notebook` while the environment is activated.

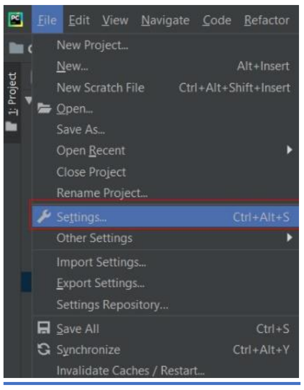


```
(deep_learn) c:\Users>jupyter notebook
```

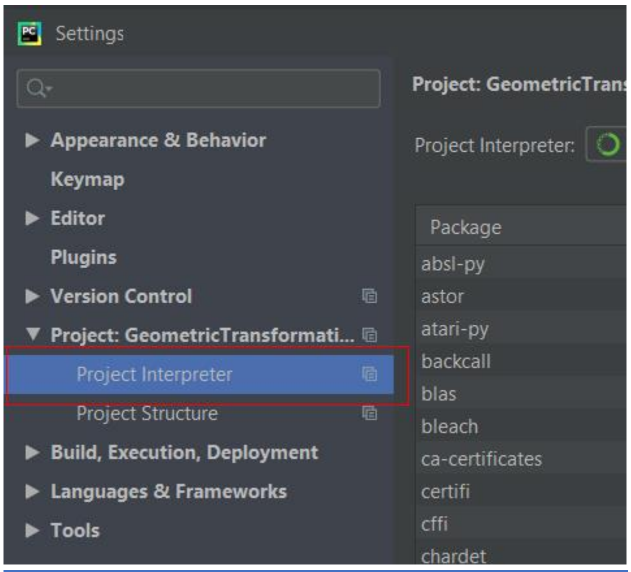
Part 2 - Setting up PyCharm to Work In The Anaconda Environment

If you want to work in a regular IDE other than the notebook, you need to set-up the IDE's interpreter to the one in the new environment you created. This is because you will want it to recognize all of the libraries you use. You can use any IDE that supports Python (such as VisualStudio—VS Code: https://code.visualstudio.com/), but we (and most of the world) recommend PyCharm. You can download PyCharm community version (free) from https://www.jetbrains.com/pycharm/. Note that as a Technion student you can get a license (1 year) for the Professional version, you just need to register with your **@campus** mail at Jetbrains's website. Once you downloaded and installed PyCharm, you need to locate the directory of the environment and pick "python.exe" as your default interpreter. Working with PyCharm will allow to debug your code the "old-fashioned" way (putting a red dot where you want the code to stop)

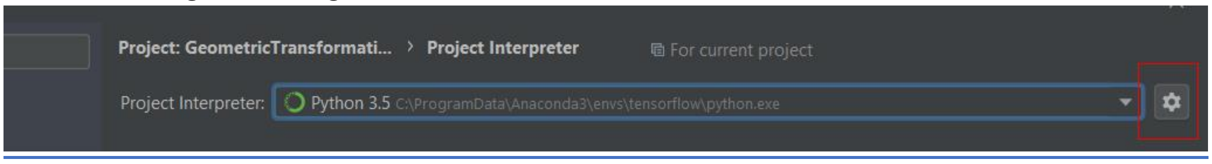
1. Open a new project in PyCharm and go to "File"->"Settings".



2. Inside the settings, navigate to "Project Interpreter".



3. Press on the cogwheel to right and choose "Add..".



Project Interpreter: Python 3.5 C:\ProgramData\Anaconda3\envs\tensorflow\python.exe

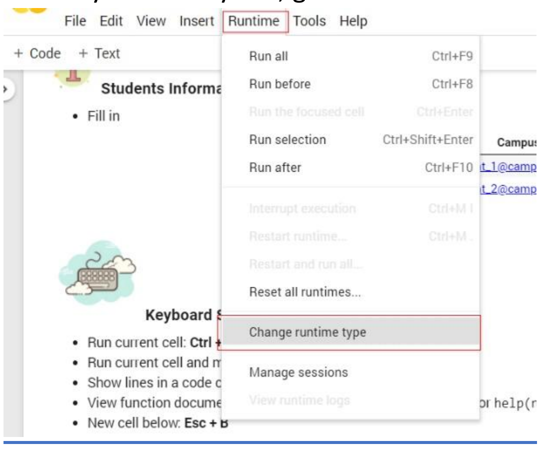
4. You will now have to find "python.exe" inside the environment directory you want to work in. It is usually in *C:\ProgramData\Anaconda3\envs\env_name\python.exe*.

5. Once done, press “Ok” and you can run your code.

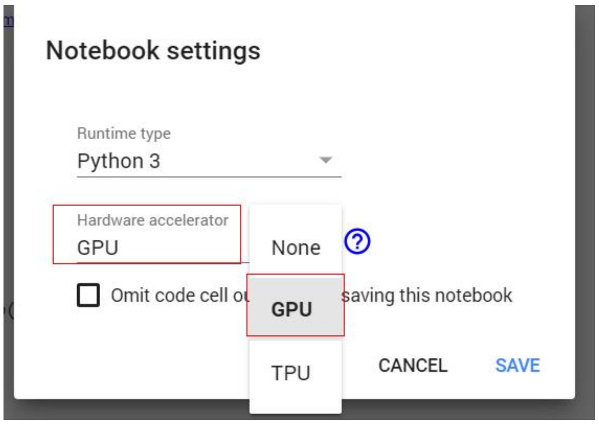
<u>Opening Jupyter Notebooks (.ipynb files)</u>: we recommend using Anaconda’s Jupter Notebook, but you can also use PyCharm (Professional version only) and Microsoft VSCode to open the notebooks locally.

PART 3 - SETTING UP GOOGLE COLAB WITH GPU

1. Go to https://colab.research.google.com, and open the notebook you want to work with (can be from GitHub, your Google Drive or just upload from your drive).
2. Before you run any cell, go to “Runtime” -> “Change runtime type”.



3. Pick GPU as the hardware acceleration type and click “Save”.

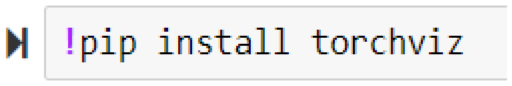

Notebook settings

Runtime type: Python 3

Hardware accelerator: GPU

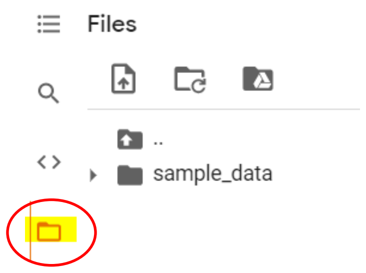
☐ Omit code cell output when saving this notebook

4. You can now run calculations on a GPU, but remember that your run is time-limited (12 hours).
5. Note the some packages are not pre-installed on Google Colab, so you will have to install manually. It is done in the same way as you would locally, but you need to add `!` before the pip command. For example, to install `torchviz` on Google Colab, run the following code cell block:

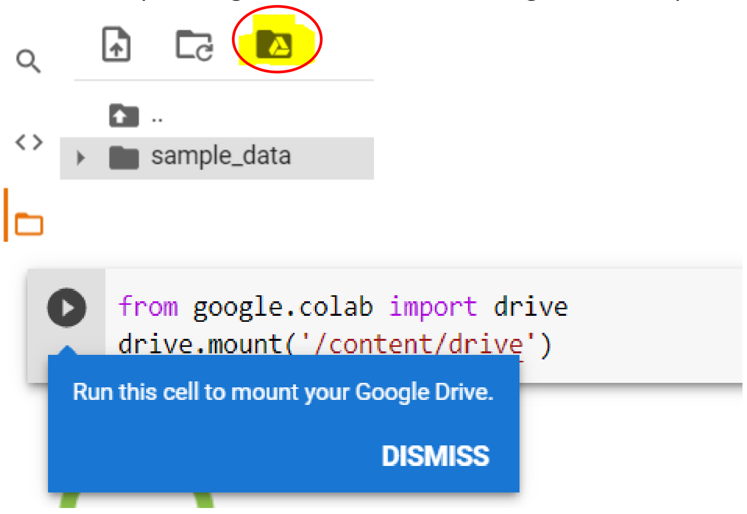


```
!pip install torchviz
```

6. In Colab, you have a file explorer that you can use to upload/download files (click on the “directory” icon on the left):



7. Alternatively, you can connect Colab to your Google Drive, making it easier to save your work and use your data (upload it once to your Drive and then connect Colab to Drive, instead of uploading the files each time using the files explorer):



```
from google.colab import drive
drive.mount('/content/drive')
```

Run this cell to mount your Google Drive.

And follow the instructions to connect to your Drive.