CITS5508 Machine Learning
Semester 1, 2022

Software Installation Guide

(drafted by Du Huynh)

If you are going to use a desktop computer in one of the labs in UWA allocated for the unit, then you can skip all the installation process described below and go straight to Section 4. If you intend to use your own laptop or your desktop at home, you will need to install *Miniconda* first. Please go through this entire software installation guide **before** carrying out your installation.

If the installation process looks too daunting for you, then maybe you should consider using *Google Colab* (see Section 6).

1 Installing Miniconda

Anaconda is an open-source distribution of the Python and R programming languages for many scientific computing applications. It helps you simplify package management and deployment. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and macOS. You don't need to install Python separately as it is included with Anaconda. By default, the latest version of Python would be installed when you install Anaconda. However, you might find that, for different projects, you will need different versions of Python.

As of January 2022, the latest version of Python is 3.10.0. However, due to some incompatibility issue with TensorFlow 2.0+, you should choose an appropriate version of Python (version 3.7, 3.8, or 3.8.5) that can be installed on your computer.

Unfortunately, Anaconda includes many packages that you don't actually need. Furthermore, we will create a specific Python environment for the unit later on, the hundreds of packges downloaded and installed in the *base* environment are unlikely to be used at all. To save disk space, you should install *miniconda* instead. It is a significantly cut-down version of Anaconda and can be downloaded from the URL below:

https://docs.conda.io/en/latest/miniconda.html

You should choose the appropriate file that is relevant to the operating system you use. Ensure that you download and install the 64-bit version[1] as most Python packages are not available for 32-bit processors these days. The installation process should be very quick.

The installation process should bring in the following executable programs: *conda*, *pip*, *pip3*, *pydoc*, *python*, etc. By default, miniconda would be installed in your home directory. However, you can also specify a directory where you want it to be installed. If you are not sure, just use the default setting for everything. For instance, if you have chosen the default setting, then after the installation, you should find the directory `miniconda3` in your home directory and under `miniconda3/bin`, you should see all the executable programs mentioned above.

You should open a terminal window and type:

```
~/miniconda3/bin/conda init
```

[1] If you are still using a 32-bit computer then you need to upgrade it. To find out whether your computer has 32-bit or 64-bit processors: (i) On the Mac and Linux, type in a terminal window: *uname -m*, if you see something like *x86_64* displayed, then it uses 64-bit processors; if you see *i686* or *i386*, then it uses 32-bit processors. (ii) On Windows, open *File Explorer*, right click *This PC* and select *Properties*. In the popped-up window, look at the description under *System type*.

This is running the `conda` program to initialize the PATH environment variable for you. From then on, for any programs you want to run, you only need to type the program name (without the path) in the terminal window that you open.

Try to reboot your computer or maybe just opening a new terminal window would be sufficient. In the terminal window, type:

```
conda env list
```

If you see the error message "*command not found.*" then it means your PATH environment variable has not been set up properly. Otherwise, you should see something like:

```
# conda environments:
#
base * /home/du/miniconda3
```

By default, *miniconda* installs a default Python environment called *base* (this is the name of the environment). The "*" symbol means that it is currently the active environment.

2 Installing TensorFlow, Jupyter, etc

Installing the Python packages needed for CITS5508 is very simple. The process is identical whether you use Windows, Linux, or macOS. The easiest way is to type the series of `conda` commands given below in a terminal window. If you are certain about the version numbers of the different packages that you want to install, you can put all the packages together by running a YAML file (e.g., the `cits5508-2022.yml` file provided for the unit).

The only problem that we encountered in past years is: some versions of Tensorflow cannot be installed or have incompatibility issues with some versions of Python and other packages. Table 1 shows the versions of *tensorflow* and *Python* that are known to be compatible. You could try installing Python 3.9 together with Tensorflow 2.5 or 2.6; however, it has been found that this caused problems to the installation of the *seaborn* package later on. So, the recommendation is Python 3.8.5 with Tensorflow 2.4.1. which have been tested on Ubuntu 16.04 LTS, Ubuntu 20.04 LTS, and macOS. If you have problems installing these versions, then try to reduce the version number of Tensorflow (but keep it at 2.0 or above) and find the version of Python that it supports.

tensorflow	Python
2.0	3.7.∗
$\geqslant$ 2.0	3.7–3.8.5
$\geqslant$ 2.5	3.9
(recommended) 2.4.1	3.8.5

Table 1: Compatibility table for *tensorflow-datasets*, *tensorflow*, and *Python*. You are suggested to install Tensorflow 2.4.1 and Python 3.8.5.

2.1 Installing the packages one by one

In your terminal window, type the following commands one by one:

1. (optional) upgrade conda to the latest version:
   `conda update -n base -c defaults conda`
   This command is required only if you had an older version of `conda` previously installed.

2. create an environment called *cits5508-2022* for the *CITS5508 Machine Learning* unit and specify to use Python 3.8.5:
   ```
   conda create --name cits5508-2022 python=3.8.5
   ```
   If you need to install an older version of Python, then replace 3.8.5 in the command above by your version number. You can replace the environment name *cits5508-2022* by any name that is meaningful to you.

3. activate the environment so that the packages installed by subsequent *conda install* commands are stored there:
   ```
   conda activate cits5508-2022
   ```
   Note that this step is very important. If you omit it, then the subsequent installation commands will put all the packages in the default *base* environment.

   Type:
   ```
   conda env list
   ```
   Now you should see the new environment listed alongside the *base* environment, for example, something like the following:

   ```
   # conda environments:
   #
   base                     /home/du/miniconda3
   cits5508-2022         *  /home/du/miniconda3/envs/cits5508-2022
   ```
   The "*" symbol should be on the line for *cits5508-2022* as it should be the active environment after we have activated it.

4. install tensorflow and tensorflow-datasets:
   ```
   conda install -c conda-forge tensorflow=2.4.1 tensorflow-datasets
   ```
   By default, tensorflow-datasets version 4.3.0 would be installed. This package is optional for the unit and is only used in the sample code for Chapter 14 from the author of our textbook. You can therefore omit it if you cannot install it.

   After each *conda install* command, you can type *conda list* to inspect the installed packages and their version numbers.

5. install scikit-learn (the latest compatible version is 1.0.1) and scikit-learn-intelex:
   ```
   conda install -c conda-forge scikit-learn scikit-learn-intelex
   ```
   scikit-learn is a very stable library and works well for almost any version of Python. We will use scikit-learn very extensively in the first half of the unit.

6. install jupyter:
   ```
   cconda install notebook jupyterlab
   ```
   Jupyter-notebook and jupyter-lab provide the interface for editing and running Python notebook (`.ipynb`) files. Both are similar and either one is sufficient for the unit. In the command above, both are installed.

7. install ipywidgets (needed in Chapter 12):
   ```
   conda install -c conda-forge ipywidgets
   ```

8. install seaborn (and matplotlib and pandas):
   ```
   conda install seaborn
   ```
   The dependencies of *seaborn* are *matplotlib* and *pandas*. Installing *seaborn* will therefore install both latter packages which are needed for the unit as well.

9. install python-graphviz and graphviz:
   ```
   conda install -c conda-forge python-graphviz graphviz
   ```

python-graphviz is for displaying *decision trees* inside Python; *graphviz* is for displaying *decision trees* outside Python.

10. install openpyxl. This backend library is needed for *pandas* to read *xls* and *xlsx* files.
    `conda install openpyxl`
11. inspect the version numbers of all the installed packages:
    `conda list`
12. cleaning up:
    `conda clean --all`

    Packages in compressed format are downloaded by each installation command above. The installer extracts them from the zip files and put them in a subdirector under your home directory or somewhere else (on macOS, it should be in `/opt/miniconda3` if you installed *miniconda*). After installation, these zip files are no longer needed. You will find that you can save a lot of disk space if you do this cleaning up step. When you are asked to confirm whether a long list of files ending with `.bz2` or `.conda` should be removed, just type *yes*.
13. (optional) deactivate the *cits5508-2022* environment:
    `conda deactivate`

    You can either deactivate the environment or just close the terminal window. Alternatively, while the *cits5508-2022* environment is still activated, you can try running some of the sample Python code provided by the author of our textbook. Note that as the library packages are installed in the *cits5508-2022* environment, whenever you need to use the installed packages you must activate the environment first; otherwise, you will see only the packages that come with the installation of *miniconda*. After deactivation, you will be back to the default *base* environment. You should see the difference by typing *conda list* here.

Some useful *conda* commands can be found in the conda cheat sheet.

2.2 Installing packages using a YAML file

To install all the packages using the supplied `cits5508-2022.yml` file, type:

```
conda env create --file cits5508-2022.yml
```

The process will take 20-30 minutes to finish. If the installation process fails at any point, you will need to sort out the compatability issues among the packages, modify the `cits5508-2022.yml` file where needed, and repeat the command above.

If the installation completes successfully, you should clean up the unwanted compressed files (see the previous subsection) by typing:

```
conda clean --all
```

To activate and deactivate the `cits5508-2022` environment, see the previous subsection.

3 Using your GPU

If your computer has a GPU (graphics processing unit) and you want to make use of it, then the installation process is a bit more complicated as it depends on what GPU you have and the version of the driver and associated libraries you have installed (or need to install). Assuming that you have a GPU from NVIDIA, you will need to have firstly installed the NVIDIA driver and then CUDA[2] and cuDNN[3]

[2] CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on its own GPUs.

[3] cuDNN is the NVIDIA CUDA Deep Neural Network library.

of appropriate versions. Below is a compatibility table to help you (for Ubuntu 18.04 or later):

	Python	cuDNN	CUDA	NVIDIA driver (for Linux x86_64)
TensorFlow 2.7	$\geqslant$ 3.9	8.1	11.2	$\geqslant$ 450.80.02
TensorFlow 2.4	3.7-3.9	8.0	11.0	$\geqslant$ 450.36.06
TensorFlow 2.3	3.5-3.8	7.6	11.0	$\geqslant$ 450.36.06
TensorFlow 2.2	3.5-3.8	7.6	10.1	$\geqslant$ 418.39
TensorFlow 2.1	2.7, 3.5-3.7	7.6	10.1	$\geqslant$ 418.39
TensorFlow 2.1	2.7, 3.3-3.7	7.4	10.1	$\geqslant$ 418.39

(see also https://www.tensorflow.org/install/gpu for more recent information)

Note that we don't use the *keras* package in the unit; instead we use the *tensorflow.keras* package, which is part of TensorFlow version 2.0 onward. So you shouldn't need to install *keras*.

Some useful installation commands are given below:

- To find out what type of graphics card you have[4], type: `sudo lshw -c display`
  If you have an NVIDIA GPU, you should see the line `Configuration: driver=nvidia latency=0` in the displayed message.
- To find out what version of NVIDIA driver you are using, type: `nvidia-smi`
- To find out what version of CUDA you are using, type: `nvcc --version`
- To install the NVIDIA driver version 450.x, type: `sudo apt install nvidia-driver-450`
- To install an appropriate version of CUDA, see the instructions on https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html
- To install an appropriate cuDNN library, see the instructions on https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html
- Finally, run all the commands given in Section 1, except for command #4 which should be replaced by: `conda install tensorflow-gpu=2.4 tensorflow-datasets` (replace the version number appropriately).

4 Running some sample notebook files

The best way to test whether your installed Python environment *cits5508-2022* works for the unit is to try running some sample notebook files from the textbook. The GitHub link is:

https://github.com/ageron/handson-ml2

Some sample notebook files are also provided on LMS. In particular, the sample file for Chapter 14 is a good one to test your installed *tensorflow* and *tesorflow-datasets* libraries work for the convolutional networks in the code.

5 High performance computing (HPC) facilities at UWA

UWA has a number of fast multi-core computers, some of which have GPUs. These computers will be available for students in this unit to run batch jobs for training deep neural networks in the second half of the unit.

- If your home desktop or laptop only has the Windows operating system installed, then you will need to install **VirtualBox** or something similar so that you can run Linux like commands to logon to the *login node* of the GPU computers.

[4] If you have the superuser (or *root*) privilege, then the word `sudo` in the commands can be removed.

- If you have a Mac computer (running macOS or OSX) or a PC that runs Linux (Ubuntu, RedHat, Fedora, etc), then you don't need to do anything.

All the CSSE lab PCs have both Windows and Linux platforms installed. When booting up a lab PC, you can select your preferred operating system to work on. Thus, you can bypass the installation of VirtualBox by using the lab computers. If you really need to install VirtualBox on your home computer, please refer to the instructions provided in the **VM_instructions.pptx** file.

Being a shared resource, it means that your submitted batch jobs will be put on a queue on one of these computers and will not be run immediately. Thus, you should always make sure that your Python code has no bugs and works correctly on your slower local computers first. More information about how to submit jobs will be given later in the semester.

Having a faster computer or having a GPU computer will help shorten the training time of your machine learning algorithms. However, it is not compulsory that you must use these HPC facilities in order to successfully complete the unit.

6 FAQs

- **What are *environments* and why do we need them?** It is useful to set up different environments for different projects when you use Python. For example, in project 1, you might need to use TensorFlow version 1.8.0, but in project 2, you might need TensorFlow 2.0. You can create an environment named `proj1` for project 1 and another environment named `proj2` for project 2. Different packages and different versions of the same packages can be installed in the two environments. You should choose environment names that are meaningful and easy to remember. For example, we use the environment name `CITS5508-2022` for all the programming work for the unit.
- **What is Jupyter Notebook?** Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualisations and narrative text. The output (graphs, plots, etc) can be displayed inside the environment. Jupyter Notebook files have the extension **`.ipynb`**. ***Markdown*** (https://en.wikipedia.org/wiki/Markdown), a lightweight markup language with plain-text-formatting syntax, is supported in Jupyter Notebook.
- **What is JupyterLab?** It is the next-generation web-based user interface for Project Jupyter. You can start *JupyterLab* by typing `jupyter-lab` in a terminal window or double-clicking on the `jupyter-lab` icon. For the CITS5508 unit, *Jupyter Notebook* is sufficient. In fact, Jupyter Notebook seems to take less time to start than JupyterLab and is easier to use. Jupyter Notebook files can be run inside JupyterLab.
- **What is Google Colab?** Google Colab (https://colab.research.google.com/) is a free web-based environment that supports the editing and running of Python programs. It has all the packages needed by the unit already installed. It also supports free GPUs. The interface is almost identical to that of Jupyter Notebook. For the later part of the unit where we need to train deep neural networks (DNNs), having access to a GPU would help you speed up the training process.

  Notes:

  – Rather than performing the installation procedure described above, you can use Google Colab for the programming work for the entire unit. Just keep in mind that Google Colab imposes an *idle time* (see https://help.clouderizer.com/en/articles/2240606-google-colab-faqs). To avoid losing your work, you should keep your session active all the time.

  – Use *Chrome* for Google Colab, as it may not work well on other web browsers.

  – As of January 2022, Google Colab uses TensorFlow 2.7.0 by default, which will work fine for the unit. However, if you prefer, you can explicitly specify which version of TensorFlow to

use (see https://colab.research.google.com/notebooks/tensorflow_version.ipynb). To find out what version of TensorFlow you are using, type the following lines in a *code* cell of Google Colab (or Jupyter Notebook or JupyterLab):

```
import tensorflow as tf
from tensorflow import keras
print('TensorFlow version: ', tf.__version__)
print('TensorFlow.keras version: ', tf.keras.__version__)
```

– From around mid 2020 onward, Google Colab randomly assigns a GPU to each user who uses the facilities for free. This means that your job may take longer to run if you are allocated an old GPU. If you can afford to pay a small monthly fee, Google Colab would be a good option and you can skip all the installation steps mentioned above.