Software Setup

This year, the recommended way to work on assignments is through Google Colaboratory. However, if you already own GPU-backed hardware and would prefer to work locally, we provide you with instructions for setting up a virtual environment.

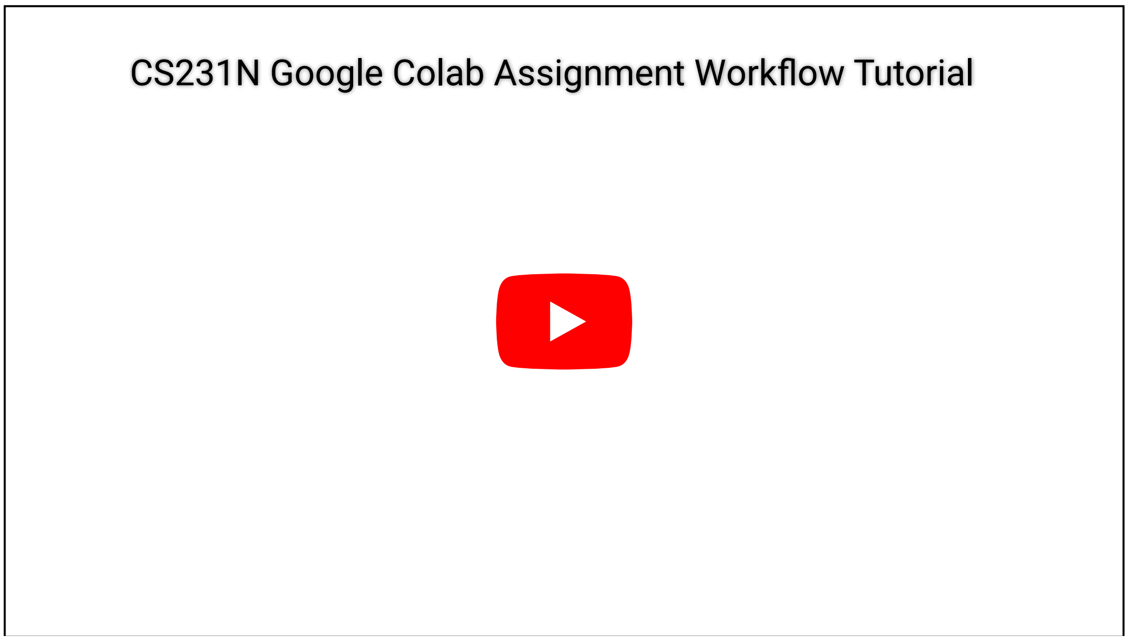
Working remotely on Google Colaboratory

Google Colaboratory is basically a combination of Jupyter notebook and Google Drive. It runs entirely in the cloud and comes preinstalled with many packages (e.g. PyTorch and Tensorflow) so everyone has access to the same dependencies. Even cooler is the fact that Colab benefits from free access to hardware accelerators like GPUs (K80, P100) and TPUs which will be particularly useful for assignments 2 and 3.

Requirements. To use Colab, you must have a Google account with an associated Google Drive. Assuming you have both, you can connect Colab to your Drive with the following steps:

1. Click the wheel in the top right corner and select `Settings`.
2. Click on the `Manage Apps` tab.
3. At the top, select `Connect more apps` which should bring up a `GSuite Marketplace` window.
4. Search for **Colab** then click `Add`.

Workflow. Every assignment provides you with a download link to a zip file containing Colab notebooks and Python starter code. You can upload the folder to Drive, open the notebooks in Colab and work on them, then save your progress back to Drive. We encourage you to watch the tutorial video below which covers the recommended workflow using assignment 1 as an example.



Best Practices. There are a few things you should be aware of when working with Colab. The first thing to note is that resources aren't guaranteed (this is the price for being free). If you are idle for a certain amount of time or your total connection time exceeds the maximum allowed time (~12 hours), the Colab VM will disconnect. This means any unsaved progress will be lost. **Thus, get into the habit of frequently saving your code whilst working on assignments.** To read more about resource limitations in Colab, read their FAQ here.

Using a GPU. Using a GPU is as simple as switching the runtime in Colab. Specifically, click `Runtime -> Change runtime type -> Hardware Accelerator -> GPU` and your Colab instance will automatically be backed by GPU compute.

If you're interested in learning more about Colab, we encourage you to visit the resources below:

- Intro to Google Colab
- Welcome to Colab
- Overview of Colab Features

Working locally on your machine

If you wish to work locally, you should use a virtual environment. You can install one via Anaconda (recommended) or via Python's native `venv` module. Ensure you are using Python 3.7 as **we are no longer supporting Python 2**.

Anaconda virtual environment

We strongly recommend using the free Anaconda Python distribution, which provides an easy way for you to handle package dependencies. Please be sure to download the Python 3 version, which currently installs Python 3.7. The neat thing about Anaconda is that it ships with

MKL optimizations by default, which means your `numpy` and `scipy` code benefit from significant speed-ups without having to change a single line of code.

Once you have Anaconda installed, it makes sense to create a virtual environment for the course. If you choose not to use a virtual environment (strongly not recommended!), it is up to you to make sure that all dependencies for the code are installed globally on your machine. To set up a virtual environment called `cs231n`, run the following in your terminal:

```
# this will create an anaconda environment
# called cs231n in 'path/to/anaconda3/envs/'
conda create -n cs231n python=3.7
```

To activate and enter the environment, run `conda activate cs231n`. To deactivate the environment, either run `conda deactivate cs231n` or exit the terminal. Note that every time you want to work on the assignment, you should rerun `conda activate cs231n`.

```
# sanity check that the path to the python
# binary matches that of the anaconda env
# after you activate it
which python
# for example, on my machine, this prints
# $ '/Users/kevin/anaconda3/envs/sci/bin/python'
```

You may refer to this page for more detailed instructions on managing virtual environments with Anaconda.

Note: If you've chosen to go the Anaconda route, you can safely skip the next section and move straight to Installing Packages.

Python venv

As of 3.3, Python natively ships with a lightweight virtual environment module called venv. Each virtual environment packages its own independent set of installed Python packages that are isolated from system-wide Python packages and runs a Python version that matches that of the binary that was used to create it. To set up a virtual environment called `cs231n`, run the following in your terminal:

```
# this will create a virtual environment
# called cs231n in your home directory
```

```
python3.7 -m venv ~/cs231n
```

To activate and enter the environment, run `source ~/cs231n/bin/activate`. To deactivate the environment, either run `deactivate` or exit the terminal. Note that every time you want to work on the assignment, you should rerun `source ~/cs231n/bin/activate`.

```
# sanity check that the path to the python
# binary matches that of the virtual env
# after you activate it
which python
# for example, on my machine, this prints
# $ '/Users/kevin/cs231n/bin/python'
```

Installing packages

Once you've **setup** and **activated** your virtual environment (via `conda` or `venv`), you should install the libraries needed to run the assignments using `pip`. To do so, run:

```
# again, ensure your virtual env (either conda or venv)
# has been activated before running the commands below
cd assignment1  # cd to the assignment directory

# install assignment dependencies.
# since the virtual env is activated,
# this pip is associated with the
# python binary of the environment
pip install -r requirements.txt
```