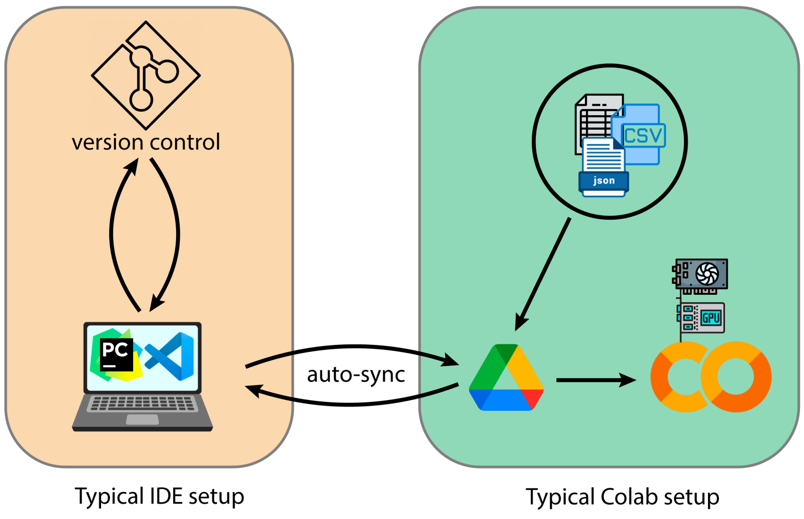
Tutorial: Advanced Colab Development Environment

David Bellamy

April 2022



0 Motivation

Most courses on deep learning make use of Google Colaboratory notebooks for their labs and assignments. These notebooks are free, extremely simple to use, integrated with Google Drive, and can connect to Google GPUs and TPUs. They run on the same interface as Jupyter notebooks, which is familiar and intuitive to many students and Python beginners.

However, notebooks do not enable good software engineering (SWE) practices and offer significantly fewer tools in comparison to integrated development environments (IDEs), like PyCharm and Visual Studio Code. These IDE tools include: debugging aids, intelligent code completion, automated refactoring (e.g. renaming a variable or file), project-wide code search, seemless integration with version control systems (e.g. Git), and support for unit testing.

On the other hand, if instructors wish to teach good SWE practices by leveraging an IDE along with their other course material, students are left to execute the code they develop on their personal computers, which introduces a serious computational bottleneck in any substantial course or project work.

As a result, courses are organized around either 1) a limited development environment with easy access to GPUs (Colab) or 2) a professional development environment with no easy access to GPUs. For (2), instructors can rent cloud compute services (e.g. AWS) for the students, but this can be costly and cumbersome to set up.

This document describes an approach that aspires to having the benefits of (1) and (2) while simultaneously avoiding their downsides. The approach is fairly simple in nature: develop code locally with your favorite IDE and version control system, set-up an auto-sync between your local project directory and Google Drive, and use Colab's computational resources to execute your project code by importing it from Google Drive.

To maximally facilitate the setup for this approach, I've written a step-by-step tutorial in section 1 and I created a template Colab notebook. The notebook contains a single cell that establishes a connection with your project contents so that you can immediately start importing your custom modules and executing your scripts on Colab resources. As this tutorial targets a fairly broad audience, it is unfortunately not entirely self-contained. In particular, it assumes a basic level of familiarity with IDEs and version control systems or the capability of navigating these details independently.

1 Step-by-step Instructions

If you are already using an IDE and a version control system for your project, you can start at step 1.2, noting my suggestion for structuring your project directory.

1.1 Initial Installations

- Install an IDE (and become acquainted).
  - The two most popular choices are PyCharm and VSCode. I personally recommend PyCharm (apply for a free student license to all JetBrains products).
  - If you are new to these IDEs, there are a ton of great introductory tutorials on YouTube for each one.
- Install Google Drive for Desktop.

1.2 Set up your project directory with your IDE and version control

- If you haven't created your project directory yet, I suggest creating a new repo on GitHub. If you already have one, I suggest refactoring it to have something like the following structure:

```
global_repo_name/
├─ data/
└─ sync_repo_name/
   ├─ __init__.py
   ├─ src/
   │  ├─ __init__.py
   │  ├─ models/
   │  ├─ train.py
   │  ├─ preprocess.py
   │  └─ utils.py
   ├─ README.md
   └─ LICENSE
```

- The idea is to place all the contents you want to have available on Colab into `sync_repo_name/` but to separate out any content that you do not want to sync with Google Drive due to bandwidth considerations (i.e. `data/`).
- Clone your project repo into your IDE so that you have a local copy of it. You will need this for step 1.3.
- Set up a Conda or virtualenv environment for your project with a recent version of Python.
- Beginner tip: make sure to add `data/` to `.gitignore` to also avoid your data syncing with your version control system. On the other hand, if you are working with small datasets, you can ignore the above points and simply sync your entire project repo.

1.3 Establish a Google Drive sync for your project directory

- Open Google Drive for Desktop and find the *Add folder* option. Select the local copy of your `sync_repo_name/` and establish an auto-sync with Drive.

At this point, you can develop code as you typically would in a local IDE environment with version control. Step 1.4 explains how you go about executing this code on Colab.

1.4 Execute your project code on Colab

- Save a copy of the template Colab notebook to your Google Drive.
- Within your copy of the notebook, connect to the desired type of Colab runtime (ex. CPU, GPU, or TPU).

- Follow the instructions in the notebook – input the path to your project directory on Google Drive, which Python version your project runs on, and (optionally) any `requirements.txt` you would like to install from (see Remarks on Python libraries) – and that's it!
  - You can now execute your project code on Colab resources.
  - Ex. `!python3.9 train.py` will execute your training script on Colab's runtime using Python version 3.9.
- **Note**: if you are idling a lot, be sure to disconnect from GPU/TPU runtimes to share those free resources with others. Getting setup in Colab is just a matter of running that one cell in the template notebook (< 1 min), so don't be afraid to disconnect and reconnect often.
  - In fact, each time you want to bring into effect any code changes that you made locally, you will re-run the template cell.

Remarks on Python libraries

If you use a package management system on your personal computer like Conda or virtualenv (and you should!), it is common to use `pip freeze > requirements.txt` to create a list of your project's required packages, and you could then install from this file in the template Colab notebook. However, Colab has 390 pre-installed Python packages and, at the time of writing, there is no way to create an isolated environment on a Colab machine – even for a single session. This means that your project's `requirements.txt` will be installed in the same environment as Colab's packages. **Beware of potential conflicts between packages and their versions**. My recommendation is to avoid using `pip freeze` on your local machine, and to instead rely on Colab's 390 packages to begin with, and to only add packages to your `requirements.txt` if they do not already come with Colab, which you will discover by getting `ModuleNotFoundError`'s in Colab for those particular packages.

Remarks on Data

To avoid syncing large datasets between your local machine and Google Drive, I recommend downloading your dataset to Google Drive and storing a separate (potentially smaller subset) copy on your local machine. Also, make sure to keep your data path abstracted from the rest of your code so that it is easy to change (ideally a one-line change in your codebase or a command-line argument you provide at runtime). That way, it is very easy to switch between local execution and Colab execution.