How to use Google Colaboratory to clone a GitHub Repository to your Google Drive?

Ashwin De Silva Follow
May 14, 2019 · 5 min read

If you are working with a deep learning project, the greatest bottleneck could be the lack of computational resources (e.g a GPU). But with Google Colaboratory (Google Colab or Colab in short), now you have the freedom to use a GPU at your disposal. Let's see how you can clone your GitHub repository into the Google Drive and run your code on top of a GPU provided by Google Colab. Here are some tips before you get in to Colab.

First, you have to get your scripts ready. For scripting, you can use an IDE such as PyCharm, Visual Code or Spyder.

When scripting, it is a good practice to properly space the codes and add comments whenever it is necessary.

You can arrange your scripts in different folders according to the tasks that they execute. (e.g : The scripts which incudes functions that carry out general utilities file handling, accuracy metrics, plotting etc, can be included in a folder called 'utilities'. The scripts which are responsible for defining the deep learning model can be included in a folder called 'model').

You can further include a 'datasets' and a 'models' folder inside your project directory. The 'Datasets' folder can be used to store the datasets while the 'models' folder can be used to save the deep learning models that you would be training. You can also have a 'logs' folder to save the log files if you are using tensorboard to monitor the training progress.

It would be convenient to save the datasets that you would be using in your deep learning project as ".hdf5" files. Within a ".hdf5" file, you can save many different datasets. For example, let's assume that you are working on a segmentation task and you have a dataset of images and the corresponding ground truth masks. You can first divide the original dataset in to a train set, a validation set and a test set and save the images and masks for

each of the train, validation and test set separately in the ".hdf5" file. This way you can have train_images, train_masks, val_images, val_masks, test_images and test_masks within a single '.hdf5' file. Having all of them under a single roof would be convenient when you load data in your code.

After the scripts and the datasets are ready, you can push the scripts to the GitHub repository. When pushing the project folder to the Git repo make sure that you do not push the dataset files (basically any large file) to it. We'll see how we can deal with the dataset files later.

After you have pushed the project folder (without the dataset files) to the GitHub repo, you can go to your google drive and create a new folder (let's call it the **project_folder**) to include your project related folders and files.

Access the project_folder. Right click on the background and select 'colaboratory' from the 'more' option in the dropdown menu that appears with the right click.

This will open a colab tab in your browser. You can change the name of the colab file in its top left corner. Next we have to connect to a GPU runtime. To do this, first go to the 'Runtime' menu in the menubar and select 'Reset All Runtimes'. Then in the same menu, select the 'Change Runtime Type' option. Select the 'GPU' option under the 'hardware accelerator' field in the dialog box that appears and click 'save'.

Now, you can click the button that says 'connect' in the top right corner of the page. This would allocate, initialize and connect you to a GPU runtime.

Since we plan to store our porject realted files in the project_folder we created earlier, we have to mount our google drive in to this runtime. In order to do that, type in the following in the first cell of your colab and run the cell.

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

This would prompt a URL with an authentication code. After you insert that authentication code in the provided space, your google drive will be mounted. You can

check the contents of the current folder in the runtime by typing the following and running the cell.

```
! ls
```

If the drive is mounted correcly, you would see that the current folder has a directory called ‘gdrive’. This is where you can find your googlde drive contents. Now, to access the project_folder we created earlier, type in the following and run the cell.

```
%cd gdrive/My Drive/project_folder
```

Since we are now in the project_folder we created earlier in the google drive, we can clone our GitHub repository inside this folder. For that, type in the following and run the cell.

```
! git clone link/to/your/repo
```

Now you have successfully cloned your repository in to the google drive. For pulling the changes from the repository, you can use the following command.

```
! git pull
```

You might want to install python libraries in the runtime to successfully execute your code. For that, you can use the following command.

```
! pip install <desired-python-library>
```

Finally, you have to upload the datasets to the drive first. It is fairly simple as all what you have to do is, to upload the dataset file inside the the ‘datasets’ folder under the

clone repository. (There is no rule dictating that you should upload it to a folder inside the cloned repo. You can upload it anywhere inside the project_folder as long it is easily referenced and acceesible)

If the dataset is downloadable as a compressed file, you can download it to your computer, uncompress it and upload the extractions to the project_folder in the google drive.

When you need to access a dataset that is stored in a GitHub repository, you can clone it inside the project_folder using the following.

```
! git clone link/to/the/dataset/repo
```

If you have an executable python file inside the cloned repo, you can run it using the following command after accessing the directory with that executable file.

```
! python <executable_file.py>
```

Or else you can execute code directly from the colab cells. But first, you will have to import the functions from your git repo that are required for the execution. For example, let's assume that you want to access a datset and the relavent functions are saved in a python file called file_handling.py under the subfolder 'utilities' in the cloned repo. To import these functions you can use the following code.

```
from utilities.file_handling import *
```

After importing the necessities, you can execute commands in the cells like you do in a python interative shell.

When the runtime restarts, the mounted drive would be dismounted. Therefore, each time you restart the runtime, you will have to mount the drive back again in the runtime. To save yourself from typing in the commands again and again, you can

include the following command bundle in a separate cell. This would mount the drive, access the repo folder and pull the changes that you committed.

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

%cd gdrive/My Drive/project_folder/cloned_repo_folder

! git pull
```

That winds up this article. Happy coding in Colab! Let me know your questions in the comments.

Github Google Colab Google Drive Clone Repository Machine Learning