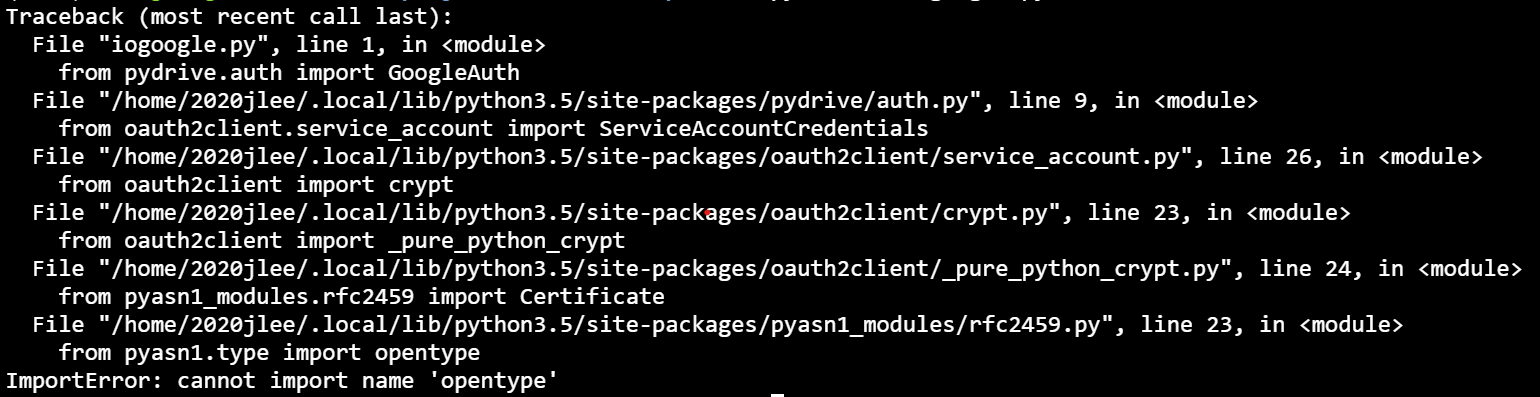
**Friday 12/13/2019**

Now that both Connor and I can connect to Google Drive remotely via PyDrive from our local computers, the next step of our project was to migrate that capability to Director. The next major concern in my mind was how to make the script run automatically (i.e. retrain once every week or only retrain upon receiving an update in the training dataset). We would like for our script to also detect when a new model has been trained and whether this model triumphs the previous best model so that it can know to download that latest trained model.

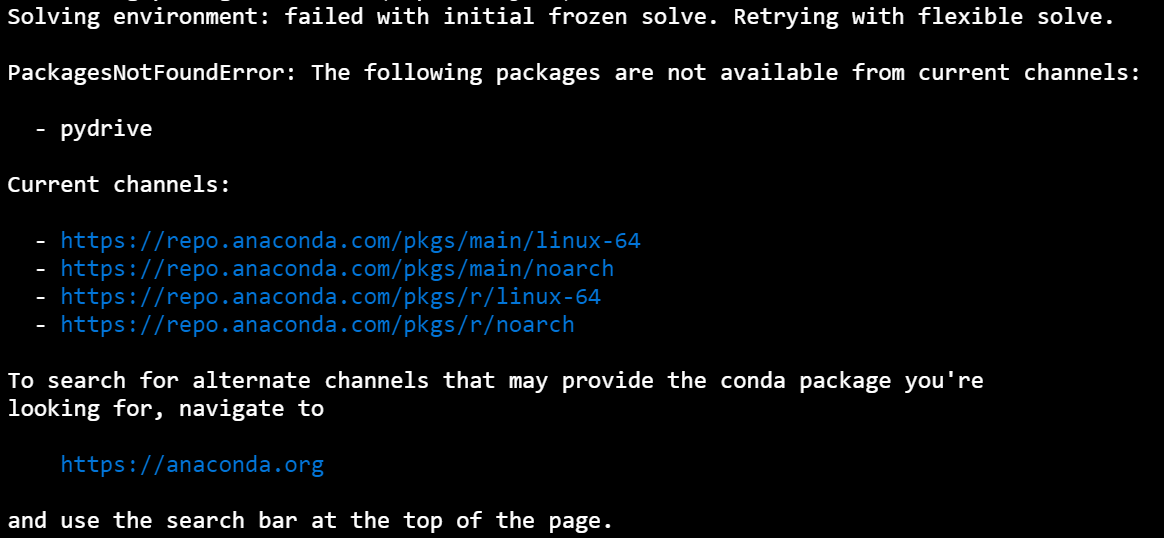
There was, of course, no problem uploading “iogoogle.py” up on Director. However, I strangely received an error of “ModuleNotFoundError: No module named ‘pydrive’” upon running “iogoogle.py” despite having done “pip install PyDrive” on the Director terminal. I even tried “pip install pydrive” just in case there was a difference between “PyDrive” and “pydrive”, but that did not work out either.

I then realized “pip” was only for Python 2.7, and it was really “pip3 install PyDrive” that I should have been doing. But nope, the terminal spat back out “requirement already satisfied…”. I sat there wondering what to do. I then happened to notice a key detail. When I did “pip3 install PyDrive”, it was storing the “PyDrive” package inside of a Python 3.5 folder, rather than Python 3.7.4 like Director was running my web application on. I tried seeing if I could run a specific version of python by doing “python3.5 [name of script].py”, and first I did “python3.5 --version” and verified if that worked. It turned out it did, as the terminal printed out “Python 3.5.2”. I then tried “python3.5 iogoogle.py” to see if anything would work better this time. But nope, while I did not get the same error, I received a different error.



Then, I realized that I should have been doing “conda install PyDrive” instead of relying on pip since conda was a key bridge to let our web application be compatible with Director. By this, I mean that pip only goes up to Python 3.5 simply because of Director itself, and how we were able to run Python 3.7.4 on Director in the first place was through Anaconda and with Mr. White’s efforts of setting our website on a particular Sys Lab machine.

However, even as I did “conda install PyDrive”, that still did not work successfully because I received an error saying:



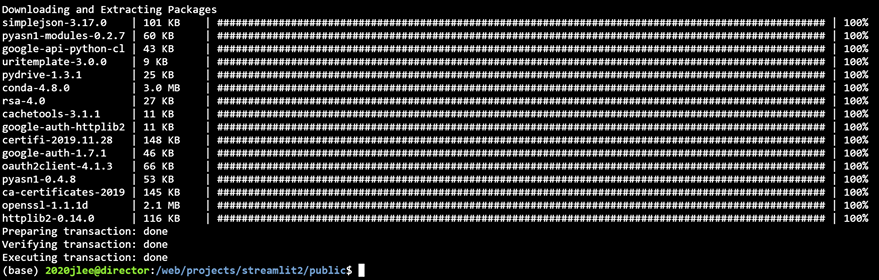
“Well, that is a real bummer,'' I thought. I then googled “how to install PyDrive on anaconda” to see if there was any possible way out. I was lucky to see the following website: <https://anaconda.org/conda-forge/pydrive>. I then followed the first of three commands to help me install PyDrive via anaconda:

“**conda install -c conda-forge pydrive**

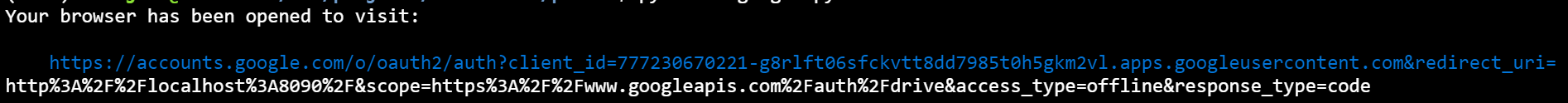
conda install -c conda-forge/label/gcc7 pydrive

conda install -c conda-forge/label/cf201901 pydrive”

It worked!



I was finally relieved once it worked, and as a result, I no longer had that “opentype” error. Instead, my terminal gave back what it was supposed to:



**Monday 12/16/2019**

Today, I heard from Connor before Sys Lab that he confirmed that the attempt to connect to Google Drive from Director was successful since he was able to remotely download export.pkl. Yay! He told me he wanted me to work on figuring out what was going on in the “colabctl” section of our project on Github. I thought what he told me was a little vague, so I asked him to please clarify.

He told me that since the code to connect Director to Google Colab was confusing, it would be nice if I could try figuring out what some of the code was doing (i.e. what is each line of code’s role, essentially) so that we could modify it to our own personal needs. Our own personal needs were that we did not want a script that would run our Google-Colab-retrain-neural-net-file to run every 30 minutes as the original script did. Instead, we wanted it such that every time one ran the “colabctl” script, it would run the Google Colab file once and only once. Essentially, we wanted a script where we controlled when to retrain our neural network, not the computer.

So, that was exactly what I attempted to do. I noticed a large portion of it depended on a package called “Selenium”, and I may need to become familiar with that package (or maybe not, depending on the intuition/description of the code to see if I could figure out what Selenium was doing without having to read the documentation itself). The code looked rather complicated: <https://github.com/TensorTom/colabctl/blob/master/colabctl.py>, but I did not want to assume much before I started digging into it. So, I dug into it.

**Wednesday 12/18/2019**

Today, I tested out the Google Colab file Connor made (which he did not test yet) that retrained the best model we have. The reason for having this file is to test that we have organized our dataset structure in our personal Google Drive accounts correctly. Recall that using FCPS accounts would cause problems with connections from PyDrive, so we had to replicate our structure over to our personal Google Drive accounts.

While the Google Colab file was working just fine, I had to think about how the saving process would work. I did not know whether to save just the best model based on accuracy or to save all the models I trained so far, just in case the best model I saved based on accuracy isn’t actually the best model and has overfitted to our data instead. It might be that a model with a pretty good accuracy -- maybe not the best -- turns out to be the best neural network. Of course, I have not quantified what “good accuracy” means, but I decided that only all neural network models we have that have a 90%+ accuracy have a chance at being our best model.

Another issue I had to consider was when a script from our Director project did connect to a Google Colab file, how was it going to verify my Google account with an access code? If I remember and correctly in Selenium, there were a couple methods that allowed you to run a Chrome application virtually, so I may consider that.