**Friday 10/4/2019**

Aha! Finally! After much research, I found this: <https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/>, an article by Pulkit Sharma that uses local repository images for their dataset for a Convolutional Neural Network setup, and has labels to go on each image. This surely will have useful code I may adopt for implementing my own. I noticed Mr. Sharma here had his images in grayscale, while I currently have mine in RGB. “In order to have any chance of implementing a CNN in the first place”, I thought, “I might as well have a separate folder where all my images are in grayscale as well.”

**Monday 10/7/2019**

My partner and I decided to divide and conquer. Essentially, what’s going to happen is that my partner is going to work on making a website that can integrate a basic neural network (based upon code from his A.I. class from last year), while I implement a better neural network using the Python Machine Learning library PyTorch. So, I started on a Google Colab file so that when I run my code, I can run on a GPU so that training time could be faster.

So, as a refresher (to whoever is reading this, whether it’d be Dr. Gabor, my 50-year-self, etc.), the way Google Colab works is that you’re essentially running your code in a virtual environment. Google has probably millions of these virtual environments that they partition to everyone who uses Google Colab. As long as you have a browser tab open on the Google Colab file, your virtual machine will still be running, keeping all local variables and files downloaded. However, it’s possible to switch to a different virtual machine if necessary, or for Google to force you to use a different virtual machine because the current machine is not working, ridding of all your temporary files stored on the virtual machine.

Therefore, to resolve this problem, I mounted my virtual machine to my Google Drive (a cloud storage system) because cloud storage is permanent; it never goes away. I then successfully made code that can read in the annotation (the .csv) files in Google Drive using the Python package pandas, navigating through the appropriate directories to get to those files using the “ls” and “cd” commands.

By this point, the time I had during class ran out, and so I decided my next plans would be to continuing following the steps of the link and making analogous steps with my code. Specifically, the next thing I would work on is looping through all the IDs of the images (0000, 0001, 0002, …) and extracting the list of pixel values for each of those images, and feed them into a PyTorch CNN.

**Wednesday 10/8/2019**

While all went successful on Monday, I started running into some problems today. What happened was that as I made a for loop to go through all the images in a certain google drive folder and tested it (and saw that it worked), I then could not navigate through directories anymore. When I tried doing “ls” for example, it gave me: “shell-init: error retrieving current directory: getcwd: cannot access parent directories: Transport endpoint is not connected

ls: cannot open directory '.': Transport endpoint is not connected”. I suppose this was because I actually stopped the for loop midway; I did not need any more re-confirmation that my code could read through all the images.

I then realized though that I forgot to upload the resized images onto Google Drive, so it took a little while to upload those folders up. Then, I decided to limit the loop to 10 iterations (instead of going through the entire folder) so that the amount of time the for loop takes would be greatly minimized. The for loop worked once more, and once I tried “ls”, I was still in my current directory without any errors.

Then, using OpenCV for Python, I had no problem extracting the pixel values from each image. When I printed the values out, however, I noticed the format was weird. It was a NumPy array of arrays like I expected, but each subarray had the same value (e.g. [128, 128, 128]), instead of different values (e.g. [128, 256, 25]). I could not figure out why that was, and I ended up stuck there by the end of class. I decided if I can not figure out why that was happening, I would have to then use some other Python packaging library (maybe PIL) to accomplish this task.