**Saturday 3/14/2020**

Hearing about how school was canceled through April 10th, I was very pleased but also kept in mind that I still had to be accountable for my research. I also had to put in three hours of research at some point, to make up last Friday’s (yesterday’s) research and the Friday before that.

I decided to put in three hours of research on this day, from 8 PM-11 PM. I refreshed my memory of what I did last Wednesday; I was making a template on Google Colab. So, I progressed along. The initial error I encountered was that after I had mounted my Google Drive onto the Colab file, I did not navigate directories correctly. path\_to\_earthquakes = Path('.') / 'google\_images' / 'earthquake'; print(path\_to\_hurricanes.exists()) returned False. To resolve this issue, I used another code snippet provided on the list found in Colab: “%cd /gdrive”.

I then wanted to check how I managed navigating through directories through another Google Colab file, where I trained a neural network using google images on Google Drive. What blocked me was that FCPS blocked the usage of Google Colab, and my partner had code to transfer over that code to personal drive. I did not recall where that code is/if he shared it with me at all, so I contacted my partner in regards to his code. Connor said for me to create a new Google Colab file on my personal drive, but then he told me it would be easier if he could handle it himself. I thanked him and patiently waited. However, then he told me that he had already made the transfer of files. I was confused because I could not find the Colab file on my personal drive, but after some conversation with my partner, I finally located it in our “Sys Lab 2” folder. I was glad I could finally witness my old code so I could know how to proceed.

On that file, I found the crucial line of code:

cd 'gdrive/My Drive/Senior Year/5th: Senior Research Project with Connor/Dataset/finished'

I copied that line of code over and changed some of the directories as appropriate. Then, I realized we did not have any image data on the personal drive. Since I did not want to have to download google images again, I migrated the template-annotations code over back to the FCPS school drive. (Recall that the reason for our project files being in our personal Google Drives in the first place is that PyDrive was not compatible with FCPS (and it was for our retrain milestone, if I remember correctly.))

However, I then realized that working on the Colab file on FCPS would not help either since the directories on personal drive and the directories of FCPS drives were different. Therefore, that would cause the retrain feature, which would use the FCPS directories, to not work. “Alright then,” I said to myself, “I will just have to download these folders and migrate them to the personal drive then, even though this process will take a while.” I chose to manually download the folders instead of PyDrive due to the fact that my partner and I were not sure if PyDrive could support the transferring of large folders anyway. We knew it could loop through every folder and transfer each individual file, but I simply thought the downloading process would take less time than coding the loop up and potentially having errors.

The process began. Then, I realized something midway. I was supposed to just upload the “first\_100s” folder instead of downloading each individual disaster folder. Whoops. So, I had to delete the folders I had already uploaded to the personal drive as necessary.

I then tried validating the path to get to the first\_100s folder, and it still proclaimed “False.” Then, I remembered that it took awhile for a Google Colab processor to take into account recently uploaded files/folders. So, I terminated the current processor and used a new one so it could retrieve/detect the first\_100s folder. Indeed, it did, and I was happy.

Then, I referenced back to the code that made the annotations template in the first place, because I needed to understand what lines of code to implement next.

As I looked through those lines of code, I realized I had to think: since I had a single first\_100s folder instead of multiple folders, each representing a different type of disaster, how was I going to account for that change in the code where I created the csv template? It was not as simple as just looping over through each disaster type because of two things:

1. I had to account for the fact that I needed to append to a csv file each time I switched to a new disaster rather than create a new one in the method where I create a .csv file
2. Now that every photo is in one folder, I had to make sure to switch the disaster type for every 100 photos. This was so that I could fill in the disaster type annotation with a “1” under the appropriate column.

I resolved the first issue by simply changing the mode of the “open()” function from “w” (writing, which I did not want since it could overwrite existing content) to “a” (appending, exactly what I needed; it creates a new file if the file is not existent yet, just like with “w.”)

For issue two, I began to realize I did not need to manually switch the disaster type for every 100 photos. Instead, I could extract the disaster type from the string that told the path of the file I was looping through. However, I also had to consider sorting the list of paths so that when I do the annotations, I would not have to go jumping back and forth between folders. Instead, I wanted to annotate one folder at a time. Within each folder, I wanted to annotate each photo chronologically (e.g. “0.jpg” to “99.jpg”.) Getting the list of paths to be sorted this way was challenging at first. Sorting it by just numerically (i.e. executing sorted(disaster\_photos, key=lambda x: int(str(x)[str(x).rfind('\_') + 1: str(x).rfind('.')])), where disaster\_photos was the list of paths) would have given me each disaster’s first photo, then each disaster’s second photo, and so forth. Sorting it by just alphabetically/using the default “sorted” Python algorithm would have given me almost what I wanted, except it would go from 0, 1, 10, 11, 12, …, 19, 2, 20, …, 29, 3, 30, ... , which also clearly screwed my desired ordering up.

My solution was to first sort the list of paths by default, since that at least got me to sort them properly by alphabetically. Then, split that list of paths into sublists that I could sort through numerically, and then concatenate those lists together. I handled the concatenation with an external variable called “to\_ret.”, and I split into sublists using a for loop and Python’s slicing technique (e.g. my\_string[i\*100: (i + 1)\*100] and “i” represents the loop counter.) The full-fledged code was:

def path\_photos\_list(path\_to\_disaster):

"""

:param path\_to\_disaster: the path to a disaster folder (e.g. “earthquake”)

:return: a list of all the photo paths/directories inside of a folder containing all the photos of a disaster

of a particular type

"""

to\_ret = []

disaster\_photos = sorted(path\_to\_disaster.glob('\*'))

for i in range(5):

dis\_pho\_slice = disaster\_photos[i\*100: (i + 1) \* 100]

dp\_slice = sorted(dis\_pho\_slice, key=lambda x: int(str(x)[str(x).rfind('\_') + 1: str(x).rfind('.')]))

to\_ret += dp\_slice

return to\_ret

**Sunday 3/15/2020**

I looked at the deliverables for Dr. Gabor by March 27th, which were:

1. A rough draft of my tjSTAR presentation
2. An updated poster

I focused on the tjSTAR presentation first. I did not know what the format should have been. I asked a Sys Lab peer how he was going to approach the presentation. He said that he was planning on a standard procedure of background, motivation, question, procedure, results, and conclusion. I concurred and went along with that format.

I then told my research partner about these deadlines coming up. He commented that his director was not requiring any concrete homework per se but he still strongly suggested for his researchers to continue pursuing their research.

Then, my next steps were to first: open my presentation on Google Slides and second: review the comments my peers gave me and follow the comments that were in regard to the content of my slides. While I could practice my speaking/presentation skills, I decided to save those for later since the deliverable of a spoken presentation was due after March 27th.

It also came across suddenly in my mind that I was supposed to get milestones done by the end of March. I realized I did not have that much time left. I was glad though that I had some flexibility of selecting my devotion to the project from now until the end of March because of this coronaviral break. Although I also realize the importance of maintaining my own health and the health of my community, so that comes first above all. I thought it was important to balance both priorities--research and health--out.

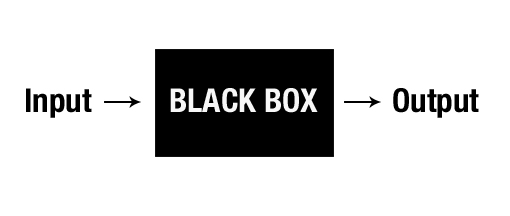
I thought that in the most realistic sense, there was probably no way to get a nearly-accurate multi-label neural network in the grounds running in my website, given the remaining amount of time I had left. It will most likely be an experimental feature set up on my website. In detail, the problems with getting an accurate multilabel neural network set up were:

1. I had a much smaller dataset compared to that my partner and I had used for our current single-label neural network that is the grounds running on our current website. For that neural network, we used 1000 photos per disaster type. For our multi-label, we had a total of 500 photos only (100 per disaster category.) While we could expand that dataset, we thought that the retrain feature could handle that part on our future web application that would include the multilabel neural network.
2. Building off of point one, those 500 photos most likely do not have all labels evenly spread out. For example, there would most likely not be equal photos of structural damage, of high vegetation, and of low vegetation as three categorical examples.

Anyway, I got back to focusing on the presentation content. I remembered that one suggestion (and this was by Dr. Gabor) was to make my slides more technical/more detailed. While my slides give a nice summary of what my project was all about, I really will need those meaty-chunky details spiced up in my future presentation recipe. Of course, from my own thinking, I would also consider the layman audience as well. I have a tendency of breaking down complicated concepts rigorously, and while I would love to do that for my tjSTAR presentation, the audience would probably be bored out of their minds a minute into my rigorous explanation. That has actually been the reason why I tried avoiding detail in my past presentations. Finding a balance between the two levels of detail is challenging, but nonetheless an intriguing challenge.

**Monday 3/16/2020**

I practiced explaining how a Convolutional Neural Network worked and did my best to ensure that even a layman could understand what I was explaining. I made analogies of the key components of neural networks a layman could understand. I decided my slide would have:



Though very simplistic, this diagram would indubitably appeal to a layman. Indeed, I would argue how a Convolutional Neural Networks works can be explained at a fundamental level. The input here is an image representing a disaster, and the blackbox is the adjustable weights (which I will refer to simply as “numbers you can finetune to better match the intended output”), and the output is a classification of the input (e.g. flooding or fire.)

In my explanation, I would break down forward feeding and backpropagation. The diagram clearly demonstrates feed forwarding pretty well. Backpropagation/the process of how a machine learns would be more complicated to explain, but the most effective analogy I came up with was improving one’s score in standardized testing, which I hoped was an experience everyone is familiar with. Essentially, I would explain that black box starts off with random numbers and will most likely have the input wrong. This is like going into a standardized test without having studied. Since a test is written by people, inherently there are going to be answers one would disagree with, but are nonetheless correct. Over time, one gets used to detecting patterns that the test writers make so that you maximize one’s chances of answering correctly, and one ends up adjusting the mechanisms one uses to answer questions. That process of learning I described here with standardized testing is like the process of fine tuning the content of the black box. A loss function describes the strength of one’s confidence in your wrong answer. Obviously, the more confident one was in his or her wrong answer, the stronger his or her mechanisms have to adjust to have his or her mindset adjust to why the correct answer is correct.

**Thursday 3/19/2020**

Today, I fine tuned certain aspects of the rough draft tjSTAR presentation. For example, I added a “General Overview” slide (as was suggested by Dr. Gabor) so that my audience can expect how my presentation will flow. I personally thought that suggestion would be beneficial, since I started off with the Civil Air Patrol and then I transitioned into A.I., which are usually two topics that are not merged together all the time.



I also looked over my peer comments. One useful comment was a confusion over the differently colored location spots in the example map that I used in the “Our Solution” slide. I had no idea either, and my sole purpose was to show what I wanted the map feature of my web application was to look like (or at least an example of what it could look like.) I did not pay attention to the different colors that much when I first inserted that image, and I became glad to remove it and pasted in the example map I displayed on my web application, which hopefully would make things less confusing.

I then added my CNN explanation slides:

