**Monday 9/16/2019**

On Monday, I focused on writing code that can take all the paths of all the photos in a certain directory and map each photo’s path to its labels. I did this through making a .csv file with the format “{insert disaster type here}\_annotations.csv” where the first column is “photo\_id” (which represents the paths of the photos), and the rest of the columns represent the labels I’ll annotate a “0” for “false” and a “1” for “true” for each photos. These labels, in a list, are: “classes = ['photo\_ID','earthquake', ‘flooding’, ‘fire’, ‘hurricane’, 'bridge', 'damaged', 'lava', 'roads', 'utilities', 'snow', 'vegetation (low)','vegetation (high)', 'building', 'flooding', 'river', 'smoke']”. So, the first couple of rows might look like:

photo\_ID, earthquake,flooding, fire, hurricane, bridge, building damage, …

google\_images\fire\_aerial\0.jpg, 0, 0, 1, 0, 0, 1, …

google\_images\fire\_aerial\1.jpg, 0, 0, 1, 0, 0, 1, …

google\_images\fire\_aerial\2.jpg, 0, 0, 1, 0, 0, 1, …

I worked on getting the paths for each image through a Python package called “pathlib”, which is useful for working with navigating, editing, and making directories. This specific package came up inside of my head because I learned it from the online Beaver Works Python course’s website, called “Python Like You Mean It” (<https://www.pythonlikeyoumeanit.com/Module5_OddsAndEnds/WorkingWithFiles.html>). The syntax for navigating through directories was very intuitive, easy, and efficient to implement, which I really liked.

Now here comes the challenging part. As I was writing code to make the .csv files, I was thinking about how I would actually go about annotating the images, because when I used pathlib’s “.glob()” method to retrieve all the photo paths, the list of all the photos did not preserve the order of the photos from left to right, and must have performed some sort of sorting algorithm when I called “sorted” on the iterative generator outputted from the “.glob()” method. Unfortunately, using the “sorted(...)” keyword was the only way to retrieve an actual list from the “.glob()” output, because otherwise I’d be stuck with that useless iterative generator as I mentioned earlier.

So, I decided it would be a good idea if I assigned a number to each image as its name instead. In addition, Connor and I just decided to stick with “.jpg” files, since that format is probably the most universal / compatible. File types like “.jpeg” or especially “.gif” might cause problems. This would provide me two benefits: not only will it preserve order, but it will also ensure that I won’t have to deal with long path names when I’m editing the .csv file for annotations. However, when I implemented code, it worked out just fine except for one peculiar issue: I would see “image0.jpg”, “image2.jpg”, “image3.jpg”, etc. and I said to myself “Ok, good. Except, where exactly is ‘image1.jpg’?” In order to possibly get some insight, I ran the code to rename the files again. Apparently, for some reason, it got the number ordering right (so “image1.jpg” ended up being there), but another weird issue was that the number of photos in the photo directory decreased from the original amount. I noticed if I kept re-running the code over and over, the number of photos eventually reduced from 842 photos to 8 photos. I was really not able to figure out what was wrong in my code because I thought my code was basic, and I did not really know how to debug my code, since the problem was not my “os.remove()” method to remove files, interestingly. This must mean there was something wrong with the “os.rename()” method section of my code, oddly enough. When I decided to share my code with Connor, however, he got an error straight away. Apparently, this was because in his dataset, there was already a photo name with the format “{insert number here}.jpg”, but through a try-catch statement, he was able to fix it. Interestingly though, Connor did not get the error of having the number of photos reduced by a certain amount each iteration of running the code like I did. I decided to give my own code one last good luck, before I decided to adopt Connor’s edit on my code. Unfortunately, there was still no luck. So, I decided I would try out Connor’s edits to my code on another day, since the day was getting pretty late at that point.

**Wednesday 9/18/2019**

During 5th period Sys Lab, certain people gave their own Sys Lab project bus pitches (equivalent to elevator pitches), and were given feedback by all peers including myself plus Dr. Gabor. I was volunteered by Dr. Gabor himself to give a bus pitch, which I had rehearsed for on Tuesday 9/17/2019. Search up “Sys Lab Project Bus Pitch” on wherever I decided to keep my FCPS schools Google Drive after graduation, because when I graduate from TJ as a senior, I’ll lose my access to Google Drive since my account will get deleted, so I have to keep all my files somewhere else.

Besides that, during 8th period on this day, Connor showed me an appealing and efficient way to annotate on the .csv file. Instead of changing some of the “0”s into “1”s on the text-file version of the .csv file (you know, what you normally think of when you hear “comma separated values” (that’s what CSV stands for.), I could pull up the .csv file onto Microsoft Excel instead. Working with Excel might be better because I won’t have to worry about accidentally changing a 0 into a 1 for the wrong label on a photo. Excel allows for a user to look at the top row values (in this case, the labels) even as the user is traversing through the bottom rows, so that a user does not have to keep scrolling back and up to see which column belongs to which label. Unfortunately though, I was not able to make time on this day to annotate any images. So, I had to stick with annotating images tomorrow.

**Thursday 9/19/2019**

On this day, I annotated 200 images: 100 images of aerial flooding photos and 100 images of aerial hurricane photos. Connor has already annotated 200 images, so in total, we annotated 400 images total. We thought that might be a good number to start off with for training our preliminary convolutional neural network, which is supposed to distinguish whether an image has a disaster or not, and at the same time, what kind of disaster it is. We plan on adding more annotated images as we go along training the neural net, since we believe maybe 400 images isn’t enough for our neural net to achieve a high accuracy. Annotating is actually something we plan on doing intermittently throughout the course of our project, actually, since the more training data, the better our neural net will be. And, if we do decide to use algorithms like YOLO (You Only Look Once), we’d have to re-annotate our images anyway because that algorithm relies on bounding boxes, which we did not decide to implement for now. We thought the binary labeling would save us time for a preliminary neural net, and it has been proven in research that one does not necessarily need bounding boxes for image classification: “[Liu et al.](https://doi.org/10.1109/ITSC.2018.8569449) demonstrated that labeling individual features with bounding boxes is not required to produce useful annotations and analytics. Instead each CAP image is labeled with semantic text labels, such as *bridge* or *vegetation*. ... Labels can then be encoded as binary values to streamline their use as a training dataset for machine learning.” (<https://github.com/mit-ll/PSIAP-CAP-Annotation/blob/master/README.md>).