**Friday 1/24/2020**

Today was particularly busy as my class and I rolled through (almost) the remaining progress presentations. It felt nice to see the fascinating projects other people were doing and to critique them as well. One presentation had some nice visual effects, where each slide in the background had a video playing automatically in the background. I thought that the usage of nice visual effects made the presentation very immersive while they also gave a clear visual sense of how the person has worked on his project so far.

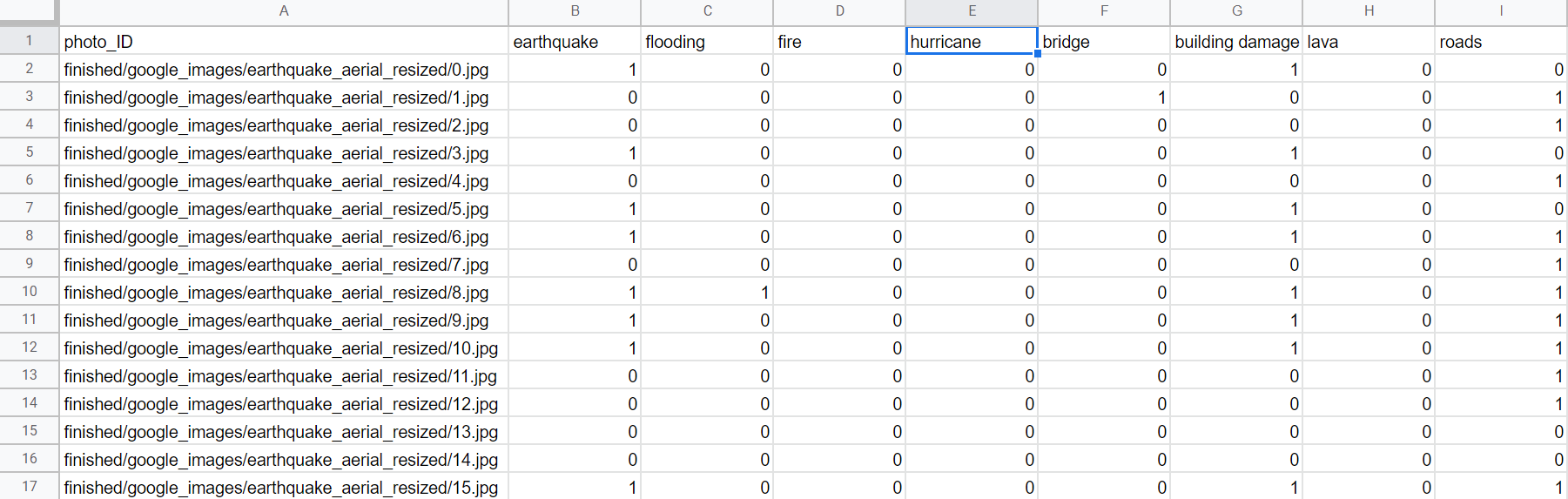
**Wednesday 1/29/2020**

We listened to the final progress presentation today, which I enjoyed. We critiqued as usual, and then I spent the remainder of class reminding myself of the progress / status of my project by reading my last journal report.

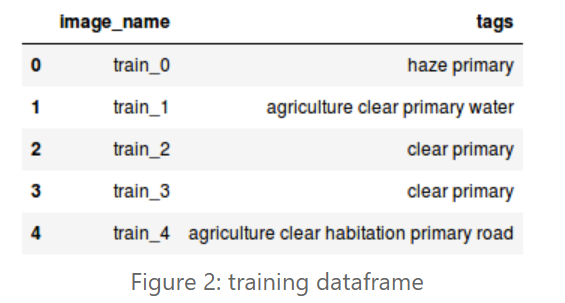
**Friday 1/31/2020**

I refreshed myself on the tutorial on training a multi-label convolutional neural network (MLCNN) from <https://gilberttanner.com/blog/fastai-multi-label-image-classification>, and on a second look-through, I understood how fast.ai processed the data in a clearer sense. Then, I realized if I wanted a strong chance of implementing a MLCNN, then I needed to finish the multi-label annotations I had started much earlier in the school year (around October ~ish).

Now, I thought about whether or not it was a good idea for me to be annotating 1000 images for each disaster type, because that would take a huge chunk of my time, as I recognized how long it took to do 100 images for each disaster type. Ah-ha! I just realized my partner and I had done around 400 images of annotation already, except for the normal disaster type. Then I realized all I had to do was to use those annotations, annotate 100 normal images, and then I would be set to go. I also realized I needed to reformat the data as well. The format of our data was:



and I needed it reformatted so that I would have the annotations as one column only as opposed to multiple columns. (Reference the image below.)

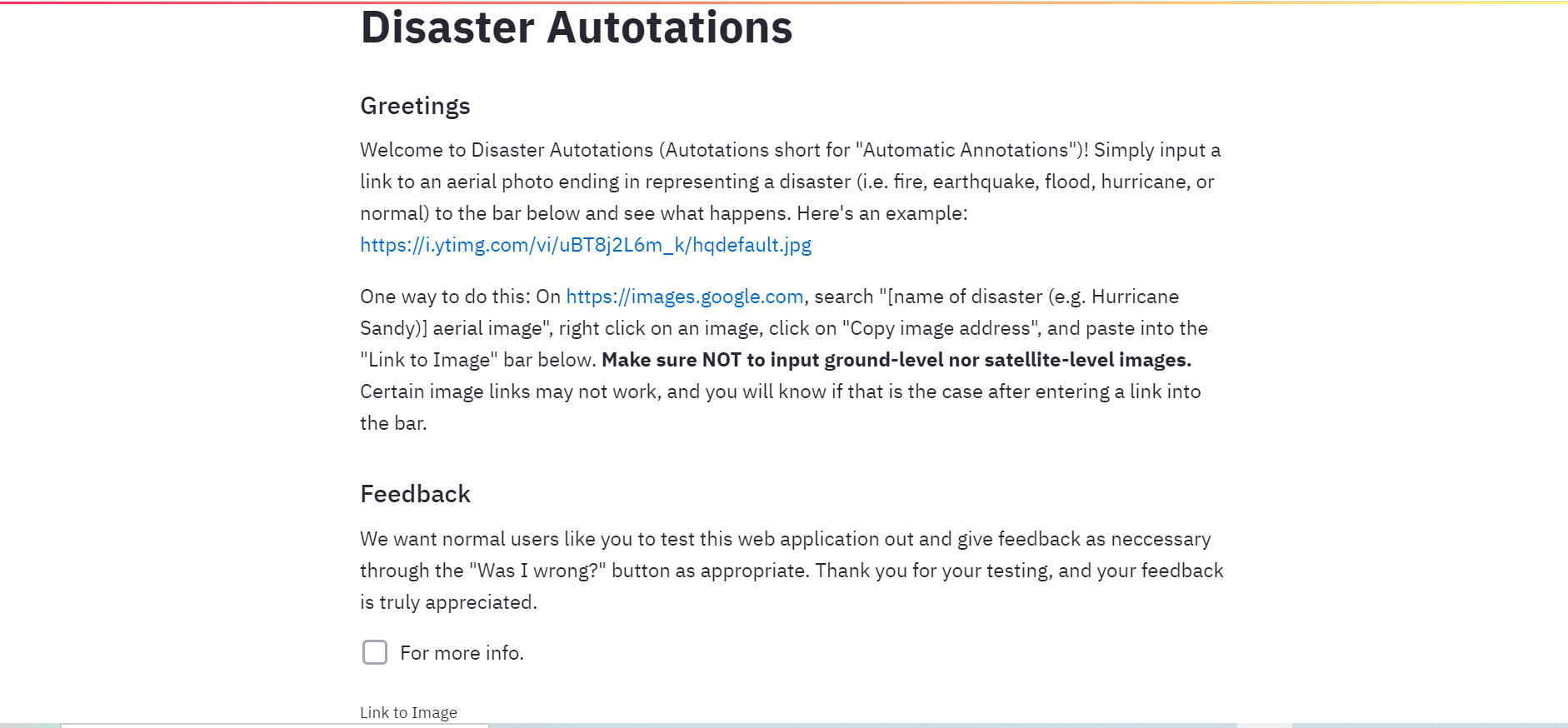


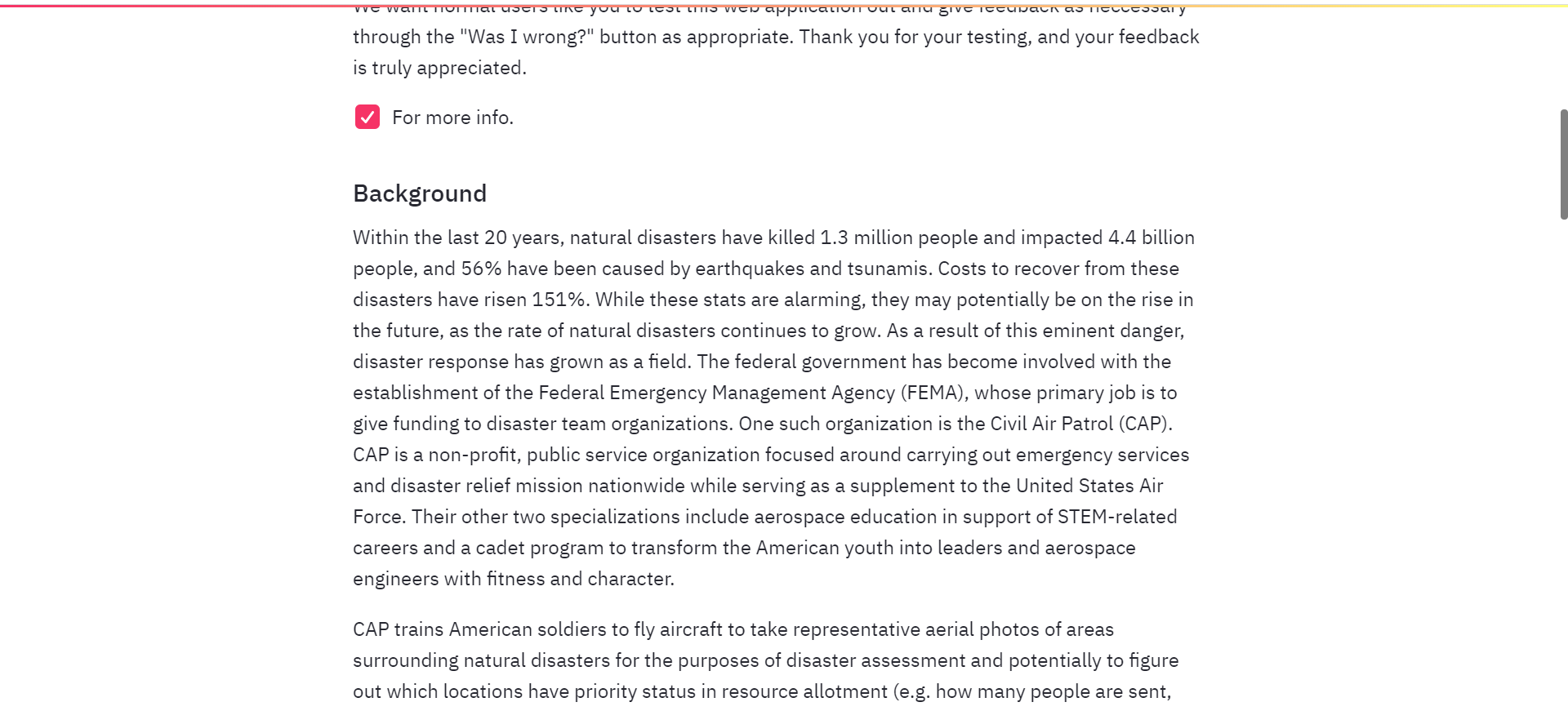
I then thought about whether there existed a method in the Pandas Python library that could accomplish making that new column, as an alternative to my having to manually reformat the annotations. In addition, I thought in the future, I would not want to manually type in annotations each time I annotated a new image (i.e. typing annotations such as “earthquake vegetation building\_damage” for every new image I would annotate would get incredibly tedious, even more so than the binary annotations I had implemented earlier.) If there is no pre-existent code to do this, then I will have to come up with code where I manually loop through each row in the .csv file and check which columns have a “1.” I, however, was against this idea as I know looping through each row without taking advantage of parallelization (where all rows can be modified simultaneously, and this is one of the huge advantages of converting .csv files to pandas dataframes in the first place) is in general, a very inefficient way to do tasks with pandas dataframes / not the pragmatic ways Pandas data frames are supposed to be handled.

I had some trouble thinking about what exactly to google to find out if there was a pandas method to do what I described above. I could go through the documentation and manually look through, but I thought surely there had to be a straightforward way to google this up.

**Saturday 2/1/2020**

Considering Dr. Gabor’s general advice on considering the user’s point of view on our project web applications, I worked on editing the user experience on streamlit2.sites.tjhsst.edu by replacing some of the text with the appropriate content I put on the poster rough draft. When I had my partner take a look, he said a user might think it would be quite interfering to have to scroll through around a page of text every time he or she wanted to test an image. He suggested we toggle the text’s appearance using Streamlit’s “button” method. I implemented a Streamlit button to hide the text that contained the “Background”, “Motivation”, “Solution / What This Is”, and “Progress and for the Future” text of the user interface. However, the problem with “st.button” was that once I clicked on it to make it show its text, I could not click the button again and make the text hidden. That was why my partner and I decided to implement a checkbox instead. That way, we could either check to show the hidden text and uncheck if we did not want to. Thanks to that Streamlit feature, our website went to looking like this:

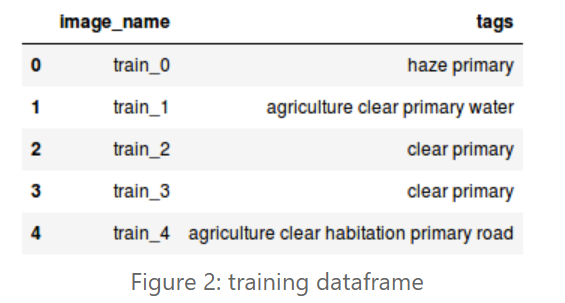




**Monday 2/3/2020**

In consideration that my research partner also has to present his progress (like I did a couple weeks ago), I decided to check over his presentation (which was essentially equivalent to mine + one more slide about our retrain feature progress). I changed a couple of words here and there (e.g. I changed the bullet point “New library: Streamlit '' to “New software: Streamlit '' so that the audience would not get confused.)

Then, I headed straight back to implementing a MLCNN. Looking at the below image again,



I realized there was another task I had to do besides simply reformatting my .csv file. I also had to get all my images under one folder rather than in multiple folders, because I also saw that implied from the sample code:

from fastai.vision import \* # import the vision module

path = Path('<path to the dataset>') # specify the path to your dataset

df = pd.read\_csv(path/'train\_v2.csv')

print(df.head())

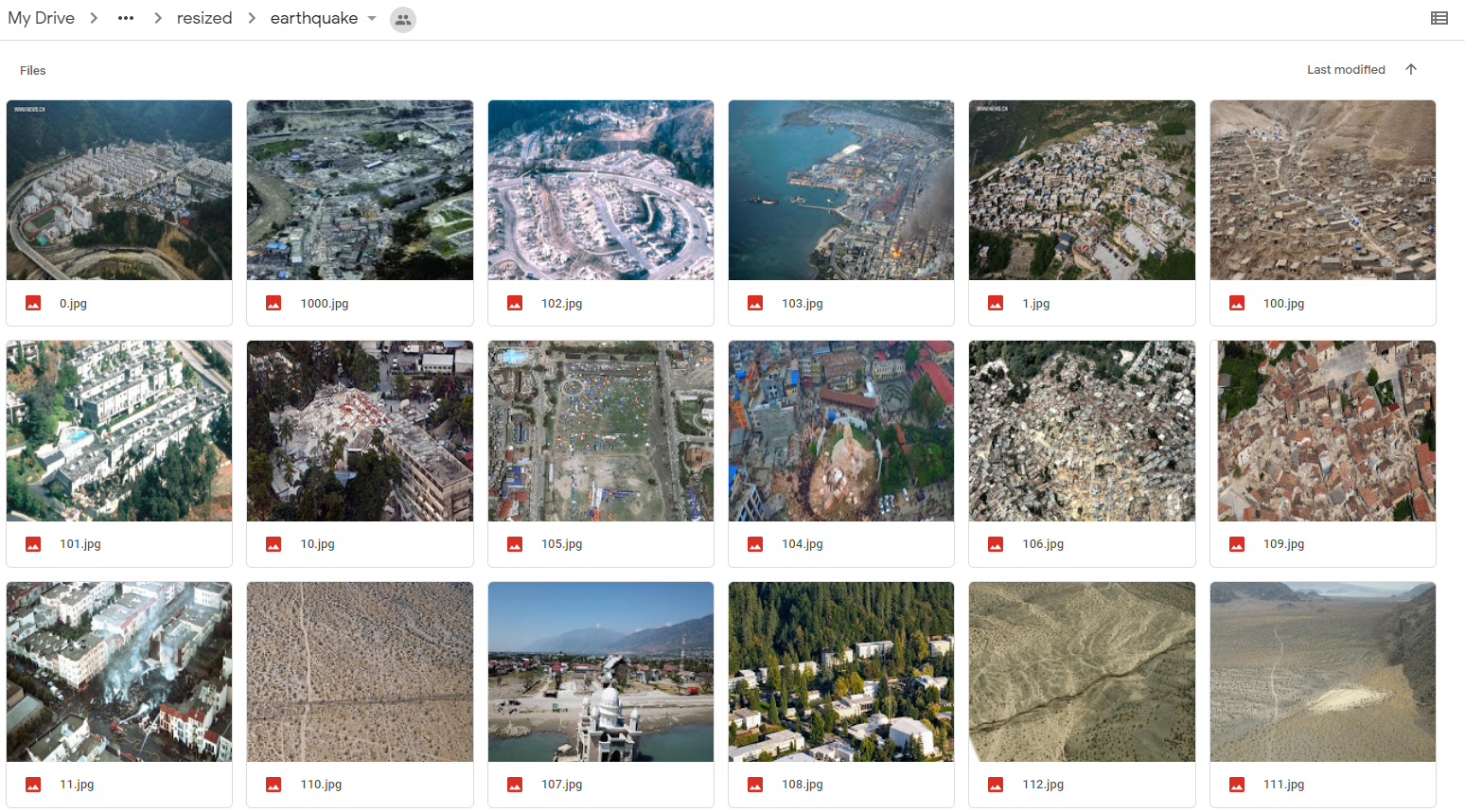
. The ‘<path to the dataset>’ seemed to be one folder in its entirety, rather than multiple folders like in mine. So, I grabbed the first 100 images in each of the disaster photo folders, and plunged them all into one dataset folder. I had the notion that each folder had 1000 images like we did for training the basic convolutional neural network I discussed earlier, but apparently I was looking at the folders that did originally have 100 images in the first place, so it was a waste of time for me to have to manually select the first 100 images and copy / paste them, when total size was just that amount.

**Wednesday 2/5/2020**

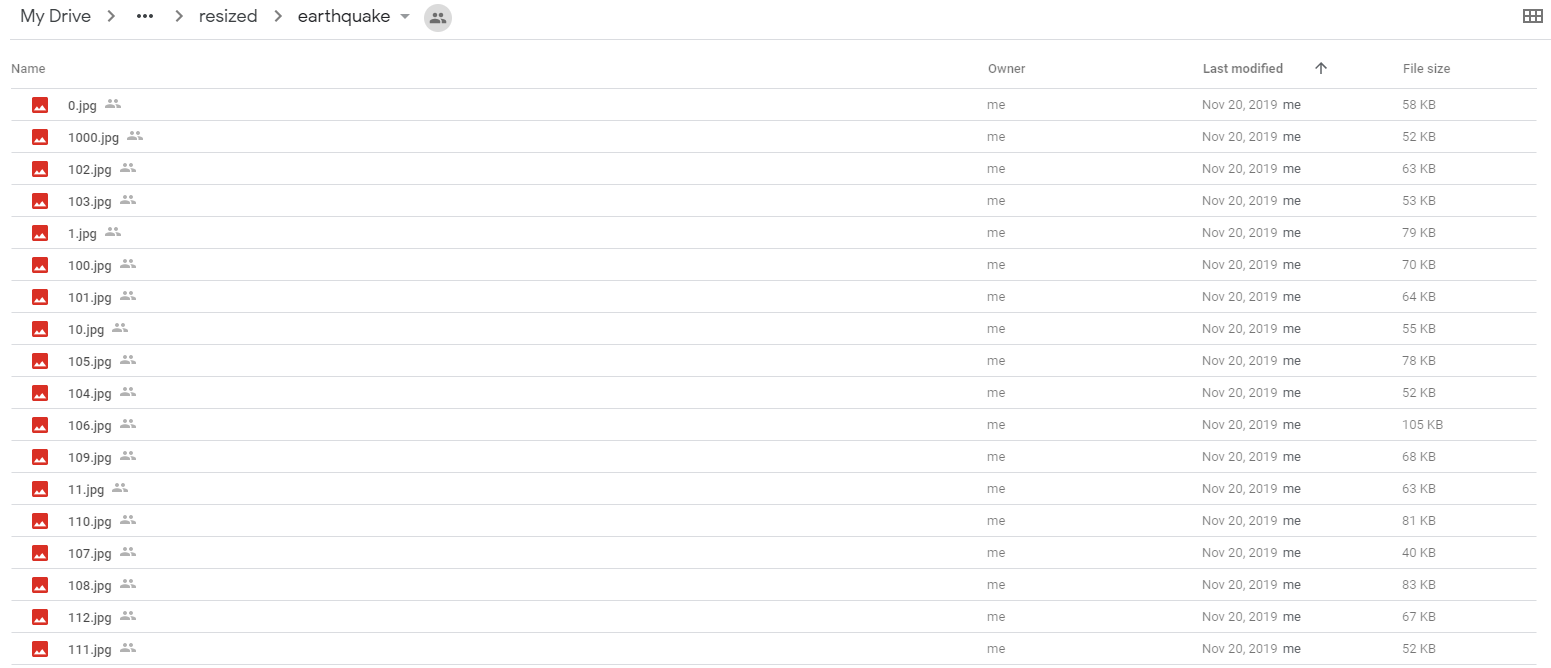
I reminded myself that Dr. Gabor wanted me to make sure all my journal reports were aligned correctly, since I submitted one journal report ahead for a while. I checked with a classmate to check which journal report I submitted at the wrong date range. It turned out to be journal report 9, since I submitted that on 11/8/2019 instead of on 11/15/2019. It took me around 5 minutes past half of class time to resolve this issue since Google Classroom was slow in handling removals of submissions of files. More specifically, for the first couple of minutes after I unsubmitted a journal report, I had “View only” access before I had editing access again. I understood that “View only” access occured in a Google document when it was submitted on Google Classroom, so I suspected there was some delay in Google handling that before I could get editing access again. I needed editing access so I could change the number of the journal report to match with the proper date ranges.

Looking back to what I said on Monday, while I still agree that I wasted time, I realized it was for the wrong reason. It was not because I could have saved time by not having to manually select all 100, but rather that all those images were not all the same size, as was required for our current CNN model. I thought I was inside the “resized” folder on Monday, but it turned out I was in a folder outside of that, which contained simply the raw images uploaded to Drive.

So, I made sure to go into the appropriate folders and copy the first 100 images from each of the disasters. I realized, however, that copying a bunch of files from one folder and pasting them into another was easy in a Windows / Mac operating system because you could control + drag your cursor. “Why could I not do something similar here in Google Drive?” I briefly thought until I realized that I had the option of seeing the contents inside each folder in list view instead. For visualization, here was the view I used earlier for copying images:



Here was the list view I could get after clicking on the top right corner (as seen in the above image).



However, the problem was that for some reason, the images were not ordered chronologically, and were ordered in some weird way because it did appear that certain images were ordered chronologically. From that point, I decided that I would need to implement a script using the “os” and “pydrive” packages that could extract images “0.jpg” to “99.jpg” chronologically from each disaster photo folder, rename each “jpg” file so that it was identifiable by disaster (e.g. “earthquake\_0.jpg” and “hurricane\_0.jpg”), and then put them into the “first\_100s” folder. I did not like having to make another script even before making a script that reformatted the data frames, which I originally intended on as my primary task. Then I thought, “Well, in the real world, I suppose you cannot have data be perfectly ordered all the time as you would like it to.”