**Friday 11/15/2019**

Today, I attempted to run Connor’s code on the school desktop so I could get a feel for the local website he has already implemented, learn the ins and outs of Streamlit myself (since it should be intuitive according to my partner), and code up what’s missing. What was missing was simply adding in the correction feature, so that the neural network could retrain itself for potentially better accuracy. Unfortunately, when I tried running on a school desktop, I received the following error message: “'latin-1' codec can't encode character '\U0001f44b' in position 3: ordinal not in range(256).”

From then on, I decided the best way to learn the ins and outs of “Streamlit” was to simply look at Connor’s code and figure out what was going on, while looking up documentation as necessary.

I had submitted the journal report #10 today on Google Classroom, but then I realized something. Last night, my partner made an important comment, but now I could not see it since I had submitted the report on Google Classroom. I then decided to temporarily unsubmit the report so that I could look at what my partner said again. Upon doing so, the comment read that he believed he found a way to make the Streamlit website accessible to the internet with the URL using a library called “Docker.” He has not tested it yet but he found it in an article, so it seemed credible to him. However, he warned that doing this would mean the website would be entirely off Director. This was the crucial link he sent: “<https://medium.com/@ansjin/how-to-create-and-deploy-data-exploration-web-app-easily-using-python-a03c4b8a1f3e>”.

Then, as I read through it, I realized it was just telling me documentation. Then, I tried looking at the documentation site itself <https://streamlit.io/docs/getting_started.html>, and I did happen to find a beginner's tutorial. The first thing it says is:

“The easiest way to learn how to use “Streamlit” is to try things out yourself. As you read through this guide, test each method ...”

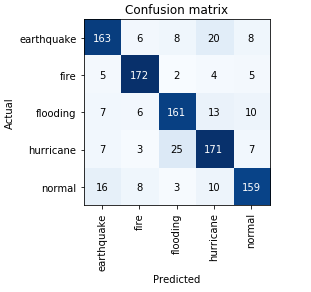
Unfortunately, I was not able to “try things out myself” at this time since I got that weird error as explained above. I looked up on Stack Overflow what this error could possibly indicate, but apparently it has to do with databases, and I had no idea how databases work and why the code involved databases, when both Connor and I clearly did not put any database code in our “Streamlit” code.

**Monday 11/18/2019 and Tuesday 11/19/2019**

I took a look at our website, and I tested a sample of test images and our neural network was giving accurate results (although realizing later the images I was feeding were in fact already part of the training dataset), but in regards to the original milestone, we forgot to make it differentiate between problem vs. no problem in an image and instead made our network classify the different sorts of problems (e.g. hurricane and flooding). So, I had to download all the google images of normal aerial photography, resize them, and put them back into the Google Drive folder. Then, I realized that the number of images for each disaster type (e.g. hurricane, flooding, and normal) should be equal for the training dataset, that way the training dataset is fair. This in turn might explain why our neural network poorly performed with a 30% error. So, I ended up downloading all google images of each type of disaster from an aerial viewpoint again, kept track of how many images I got for each disaster, and then decided on a good number of images to have for each disaster. Each number of google images for a disaster was a couple hundred above 1000, so I set 1000 to be the number of images for each disaster for the training dataset.

I then got onto the “gdrive\_fastai\_CNN” Google Colab file (the file that was copied over from Connor’s original neural network file and its goal to connect with Google Drive instead of URL files) and retrained the convolutional neural network, relieved that it would be training under a much fairer training dataset now. After the first five or six epochs of training, the error dropped from around 35% to 30%, which was really nice. It then dropped down to 26.9% around the 11th epoch, but then sadly jumped up to 28% on the 12th. “Hopefully just a minor fluctuation”, I thought. I recalled learning from the TJ Machine Learning Club about the phenomenon of underperforming (where a neural network will end up with a high error rate even after continual training), so I became a little worried temporarily, but let that brush off my mind as the next epoch error rate dropped to 27.3%, and then to 27.1% for the next one after.

I ended up with a final error loss of around 17.4% for the night because my model kept fluctuating between 17 and 18%. To get some insight as to how my model was performing, I plotted a confusion matrix as shown below:

****

The blueness of each square indicates the magnitude of the fraction of images classified amongst all possible image classifications for a particular actual class. The bluer it is, the more that square represents what the model would most likely predict for a given image of a particular disaster type. Since the bluest squares line up in a negative-slope diagonal (earthquake lines with earthquake, fire lines with fire, etc.), it indicates our model is training in the right direction (e.g. model predicts earthquake 163 / (163 + 6 + 8 + 20 + 8) = 79.5% of the time for earthquake photos, and similar ratios for the other types of disasters).

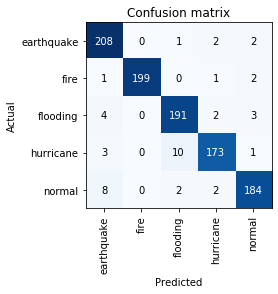
**Wednesday 11/20/2019**

Phew. I had quite a night last night for research. I was very happy about my productivity and the progress I was able to make last night. My partner, Connor, also happened to stay as up as I did and answered any questions that came along, such as clarity on the web application code, where to put the “.pkl” and “.sth” file essential for savings the results of our network architecture, and what a good error rate to shoot for in the initial training of the network I did last night. We went with 15%, but then saw how it kept fluctuating between 17% - 18%, and once we saw the network go from 18% to 17.4%, we decided that was the best time to save the neural net there instead so that we wouldn’t risk pressing our luck (luck = the neural network decreasing its error rate in the next epoch, that is).

Today, my goal was to use the “.sth” file to retrieve back the neural network I saved last night. To clarify, if I needed to retrain my neural network for a later session, I would load the “.sth” file, but if I thought my neural network was ready, I would use the “.pkl” file to use the neural net on our “Streamlit” web application. To clarify, the “.sth” file requires the dataset and recreating the “learner = cnn\_learner(...)” variable while the “.pkl” file is not retrainable and does not need the dataset.

The “.pth” file successfully loaded the convolutional neural network architecture. Hurray! As I trained my neural network even more (with a goal in mind of realistically 15% or less just for today’s class), I then decided to look up tutorials on Python flask for Director.

What the… apparently after the first epoch of training (which actually took longer for the school computer run, 7 min. 45 seconds compared to the 1 min. 34 seconds I was getting at home, even though it’s Google Colab), the error rate dropped extremely significantly. It went down from an error of 17.4% to an error of 4.40%! “Wow.” I initially thought, but then thought, “Well, this certainly seems too good to be true.” I then plotted the confusion matrix, and here were the results:



**Friday 11/22/2019**

Today, I planned on John (a sys lab administrator I mentioned in a previous lab report) looking over more in depth whether Streamlit could be compatible with Director, and I said to John on Wednesday that I would show him a demo of the code to get a real feel for Streamlit. However, I ran into one major issue: the website stated it did not find a package named “torch.” So, I installed torch. Next, it gave a problem about “fastai”, and so I installed “fastai.” However, it said it could not run “fastai.vision.” However, even when I installed “fastai” correctly, the website then produced an error “no module named fastai.vision.”

I then tried locally, and then I realized upon running the web app I had that weird error. Then I realized maybe that error was saying in clearer terms that I forgot to set up the directory path right to the “.pkl” file and I needed to get that “.pkl” file too. That resolved the issue, but now when I access the local link, it gives me an error that it could not find a package named torch. So, I tried installing torch, but the problem then was that since Sys Lab computers only allow 4 gigabytes of quota (which is essentially memory size), I did not have enough room to install torch, even when I deleted other files on the Sys Lab computer that I had backed up. From then on, I decided the best way to progress with my project is to borrow my mother’s laptop (and my mother does not mind), and work on it.