**10/11/2019**

I had two goals for today. One goal was to figure out why when NumPy processes the images, the subarray values are the same. The other goal was to continue with this link: <https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/>.

It apparently turns out not all of the subarrays consist of the same value like [ 47 47 47] or [254 254 254]. Some arrays seemed to be fine (e.g. [159 175 211]), so I assumed that some of the images had regions of the same color. I decided to confirm this by checking the actual images my loop was passing through. The first three images were white and black, and that is consistent with the [47,47,47] and other subarrays that had consistent values. I decided to not think this was a problem at all and progressed on. I copied and pasted the part of the code on the website that would create “train\_x”, “train\_y”, “test\_x”, and “test\_y” (variables needed for training and testing the Convolutional Neural Net later on). However, when I ran that code, I received a strange error that should not have existed: “FileNotFoundError: No such file: '/gdrive/My Drive/Senior Year/5th: Senior Research Project with Connor/Dataset/finished/google\_images/flooding\_aerial\_resized/0000.jpg'”. I sat there, utterly confused. I had made extra measures that all my directories were set up properly, and I double-checked to confirm that indeed that directory, '/gdrive/My Drive/Senior Year/5th: Senior Research Project with Connor/Dataset/finished/google\_images/flooding\_aerial\_resized/0000.jpg' exists within my Google Drive. Unfortunately, by this point, 5th period sys lab ended, and I had to move on with other classes.

**10/15/2019**

With a fresh pair of eyes today, I decided to progress onward with the Google Colab file for my PyTorch neural network, investigating why in the world I had that strange error I mentioned on Friday.

Oh goodness, I realized that the “flooding\_aerial\_resized” folder was not in the directory after all (when it should have been). Darn. I then selected an existing folder, “hurricane\_aerial\_resized.” Hurray, the loop started to work. However, I started to realize something. I noticed that in the tutorial, instead of doing binary classification, the author was separating each class as its own number. For example, a “0” represented “T-shirt/top” and a “7” represented “sneaker/top.” He wanted a neural network to select out of nine total classes. I realized the way I’ve been doing annotations was to list each variable (e.g. “earthquake”, “flooding”, etc.) and label them as either “0” or “1”, instead of saying “0” was earthquake and “2” was “flooding” for example.

I decided the best way to handle this was to simply come up with a “disaster\_type” column in my current annotations dataframe and label “0” for an earthquake, a “1” for a hurricane, a “2” for a fire, and a “3” for flooding.

**10/18/2019 (morning)**

I had taken a nap earlier, so I worked on research throughout the night and into the early morning. I continued along with the tutorial as usual, but I noticed a detail that was off. That detail was that in our training labeled dataset, the shape -- the dimensions -- was off. The shape was supposed to be (x, 500, 500) where “x” represents the total number of images in our dataset and (500, 500) represents the dimensions of each image. I say “x” instead of an actual number because our dataset is going to keep increasing throughout the project. However, instead of that shape, our code produced a shape of “(x,)” and I wondered why (500, 500) did not follow.

The tutorial showed code to display some of the images using the matplotlib module, and I copied and pasted that into my Google Colab file. When I ran that section of code, the images printed out all right. I was wondering whether things have just changed since the author wrote the tutorial and my computer is deciding to print “(x,)” on the outside, when internally it is “(x, 500, 500).” I double-checked this though by selecting the first element of “train\_x” (which had all the images inside) and printed it out. I saw that it printed out a NumPy array. I thought then that maybe the “(x,)” represented x numpy arrays. I ran some basic testing code on “ipython” program (essentially Python IDLE but nicer-looking) on the terminal, where I set the following:

A = []

B = np.array([1, 2, 3])

C = np.array([4, 5, 6])

D = np.array([7, 8, 9])

A.append(B) => .append C and D as well

A = np.array(A), and I expected A.shape to print “(3,)”, but that did not happen. Instead, it printed “(3, 3)” like how it should be.

“Hmmm… this is rather strange”, I thought. I then checked the shape of each numpy array inside of “train\_x” and noticed different shapes, instead of a consistent (500, 500) shape. I wondered “Wait, did we not resize all the images a long time ago?”. Then, I realized my partner and I have not. We forgot to resize all the earthquake photos and flooding photos. Afterward, I made a script to simply make resized images within a given directory and put all those resized images into another directory that I would upload on Google Drive later. Of course, I set the resizing to 500 pixels x 500 pixels, which is what I meant by the (500, 500) shape I talked about just earlier.

After uploading the folder of resized images, I then went back into Google Colab and reran the code, and I was happy to see that “train\_x” finally had the shape (361, 500, 500). Then, I progressed through the tutorial and finally was able to train a neural net, until I got this error:

RuntimeError: size mismatch, m1: [324 x 62500], m2: [196 x 10] at /pytorch/aten/src/THC/generic/THCTensorMathBlas.cu:273.

It was 7:01 AM at that point and I had to pack up and get ready to leave to walk to the bus stop. Too bad I could not get a neural net to train quite yet, but hopefully I can get it trained during this weekend, now that the only thing I have to do left is to debug this issue.

**10/18/2019 (class time)**

Stemming off with my last journal report, I received the following error:

---------------------------------------------------------------------------

RuntimeError Traceback (most recent call last)

<ipython-input-25-2335eb460d1e> in <module>()

**6** # training the model

**7** for epoch in range(n\_epochs):

----> 8 train(epoch)

/usr/local/lib/python3.6/dist-packages/torch/nn/functional.py in linear(input, weight, bias)

**1367** if input.dim() == 2 and bias is not None:

**1368** # fused op is marginally faster

-> 1369 ret = torch.addmm(bias, input, weight.t())

**1370** else:

**1371** output = input.matmul(weight.t())

RuntimeError: size mismatch, m1: [324 x 62500], m2: [196 x 10] at /pytorch/aten/src/THC/generic/THCTensorMathBlas.cu:273

Based on my intuition of PyTorch after having used it for the circle lab in A.I. class, it would seem the dimensions of the input that the neural net was expecting did not match the dimensions of “train\_x”. In addition, because I simply copied and pasted the code to build the neural net from the tutorial I was using, I think my own intuition at least near correct. In support of my intuition, the images that the tutorial was using had dimensions of 28 pixels x 28 pixels, while I was using images that were 500 pixels x 500 pixels. Therefore, I decided the best way to figure out what was going on in this runtime error, I researched how Convolutional Neural Nets were constructed using PyTorch, so I could understand the code I copied and pasted line by line.

**10/21/2019**

I had the feeling that the reason for the runtime error was that the tutorial was incompatible with Google Colab somehow, and perhaps all code was supposed to be run on a local machine.

I ended up getting a strange error with the block code right after the import statements, which told me to check the cuda version of my computer. I wonder if this meant my laptop was not compatible with CUDA GPUs, so I checked the graphics card myself. Yep, I confirmed there was no support.

**10/23/2019**

So, I continued copying / pasting the code onto my local machine to see if things would work any better, but no luck there. Analyzing part of the error carefully, “m1: [324 x 62500], m2: [196 x 10]”, I noticed that [196 x 10] corresponded with the self.linear\_layers = Sequential(Linear(4 \* 7 \* 7, 10)). 4 \* 7 \* 7 = 196 and 10 = 10. Ok, that explains “m2”, but what about “m1: [324 x 62500]?” I recognized the “324” since that was the number of images in the training dataset, but I had no idea where the “62500” came from. I was expecting it to be “m1: [324, 250000]” since my images were 500 x 500, but 62500 = (500 \* 500) / 4. I wondered if there was a reason for the division by 4. Perhaps it came from a line code of defining “self.cnn\_layers”: “BatchNorm2d(4)”? I decided to play around with the numbers here. “What happens if I changed it to BatchNorm2d(2)?”, I wondered. Well, another error occurs. “RuntimeError: running\_mean should contain 4 elements not 2”.

“Okay, clearly I will not get very far if I keep playing with the numbers and have no idea what they mean.” I then decided to investigate how Convo. Neural Net. architectures were defined in PyTorch. <https://blog.algorithmia.com/convolutional-neural-nets-in-pytorch> looked promising.

**10/24/2019**

Just like the last milestone, my partner and I were worried we were not making tangible progress. My partner decided to, during class on Thursday, see if there was an alternative. We had both forgotten about the fast.ai tutorial (and in a previous journal report, I have tried this, but could not get it to work due to my confusion with working with the directories), but he was able to get a neural net trained. We were finally able to have a baseline neural net, but simultaneously though, we were both frustrated that the PyTorch NN we have been working on for the past couple weeks still did not work. While we did decide on PyTorch based on a tutorial and my small exposure to it last year in A.I. -- the Unit Circle Lab, we came to know that PyTorch is industry-leading in research, making my partner and I less tempted to give up on the PyTorch work we have been working on so far.

Therefore, we decided to go with the neural net trained from fast.ai for now, and put that on the website, and use PyTorch later for multi-label classification, a prime later milestone. Honestly, we would check through all other libraries (e.g. TensorFlow and Keras) to see if they perform any better.