**Friday 9/27/2019**

Today, I adopted Connor’s code to see if the code to make a new folder of images that represented the resizing of each of the images within another folder. Apparently, he commented in the code that that linux / Mac systems may have a potential problem since “WindowsPath()” objects do not work. However, I had no problems when I ran the code itself. However, I will touch base with Connor just to see what he was commenting about after class today. I wanted to check out what the resized images, and they actually, upon some glances, do not look bad at all! Resizing images should have been something I was open to, instead of trying to force myself to implement a concept outside my bounds: a dynamic neural network.

Praveen, my neighbor, found a site on Director that showed how to make dynamic websites. I asked him “Hey Praveen, where exactly did you find that?”. Eventually I was able to enter the link he was on, and I happily saw the guide on making dynamic websites. I assumed that dynamic websites were essentially web applications, in contrast to normal websites, which are essentially internet pages with mainly textual information. This is the link if I need it: <https://director.tjhsst.edu/docs/dynamic-sites-on-director/>

And, if I just want to learn how to make web servers in general: <https://director.tjhsst.edu/docs/> would work. I believe this link will be useful for Connor and I to learn how to make our final product, since we both do not know web application development yet.

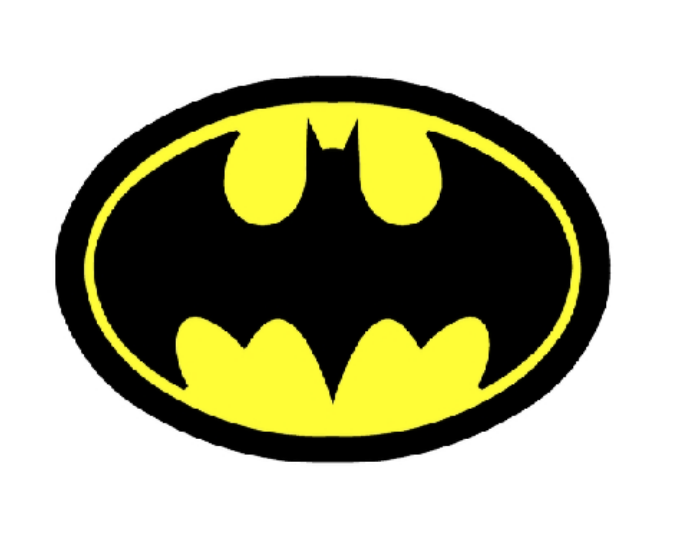
Anyway, back to the real picture here: I want to first off, create a convolutional neural network that can train on images and their manually pre-determined labels as input, and then test it with new images, and will hopefully classify whether there’s a disaster or not by indicating a “1” on “earthquake”, “flooding”, “hurricane”, or “fire”.

It was a study hall 6th period, and I decided to use that time to look for a sample implementation of a Convolutional Neural Network. I looked at this website: <https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-satellite-photos-of-the-amazon-rainforest/>, which attempted to classify satellite Amazon rainforest photos into one of 17 classes, some of which were “agriculture”, “water”, and “clear”. My goal was to read through the website and see if I could get an idea as to how to implement my own Convolutional Neural Network.

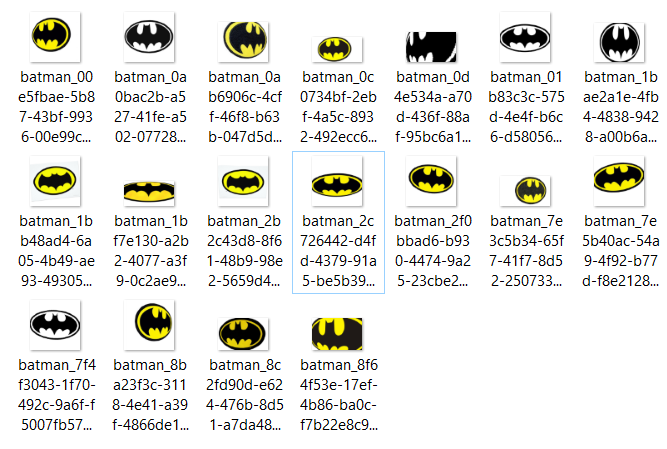
Hmm… an important note from the website is: “The problem is an example of a [multi-label image classification task](https://en.wikipedia.org/wiki/Multi-label_classification), where one or more class labels must be predicted for each label. This is different from multi-class classification, where each image is assigned one from among many classes.”. I realized then that what I’m looking for is actually multi-class classification, because among the disasters “flooding”, “hurricane”, “fire”, “earthquake”, or “normal”, I want obviously the neural network to classify each image as only one of those labels. Sure, you may argue that certain disaster can overlap (e.g. a fire and earthquake could be present within an image), but I want to avoid the possibility of my NN classifying an image as both “normal” and “flooding” for example. I believe multi-class classification is the way to go. Multi-label classification though something I intend to implement in a neural network later, but for now, Connor and I are just looking at: “Is there a problem or not? If there is a problem, what kind of problem is it?”

I looked at this link, which addressed Convolutional Neural Networks with multi-class classification: <https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f>. I noticed it was using the MNIST handwritten numbers training dataset. Then a flashback came upon me: “Wait a minute, did I not build a convolutional neural network to classify MNIST handwritten digits before, during the TJ Machine Learning club session after school?” What I decided to do next was to see that code I made during that session.

But then, I realized that the data was extracted from online, not from a local repository like I have my image dataset in. However, this link seemed more promising since it does address folders of images: <https://medium.com/@ipaar3/how-i-built-a-convolutional-image-classifier-using-tensorflow-from-scratch-f852c34e1c95>. Yay! In addition, this link provides a link to a Python application called “Augmentor” (<https://github.com/mdbloice/Augmentor>), which can increase the image training dataset you have already by making multiple copies of each image at different angles. For example, if I had a picture of the Batman logo (to make a NN that can differentiate between Batman and Superman):



Augmentor would allow this:



^^ and that’s only for one image. Imagine how much larger our only-400 image dataset could increase if Augmentor can produce that many copies per image. Of course, you want to avoid an upside-down transformation of an image, because that is obviously false representation. Humans do not experience the world upside down.

**Monday 9/30/2019**

Connor and I worked on code to be able to take in a .csv annotation file as input, process each row, and append all RGB-pixel values in row-major order to that row in order to have them readily accessible as input to our eventual convolutional neural network. We used packages like “orderedDict”, “csv.DictWriter” and “csv.DictReader” to assist us with this process. On this day though, we did not get a chance to actually test and run the code, so we just uploaded it to keep it in a safe place.

**Wednesday 10/2/2019**

We then tested our code and saw that it did not work. I was initially confused because somehow the path to read in the image was not found. I did not understand why this error occurred, as I clearly had the correct path for each image. I decided to look into the “orderedDict” object we made, since the error seemed to be blabbing on about that. The “OrderedDict” seemed to say {“labelid”: “label\_id”, label1: label1, label2: label2, …}. I thought this might have been just a strange format of how the “OrderedDict” normally behaves, but then I realized how “OrderedDict” actually worked. It tried storing the name ( the label\_id, feature1, feature2, etc.) of each row as keys and the actual values of each row as output. I realized then somehow the code needed to skip over the first row. I tried adding a “[1:]” like I did when I would loop over lists in Python, but sadly that did not work either. Obviously, I could not get rid of the first row entirely, because of course then, the computer would not be able to extract “label\_id, feature1, feature2, etc.” in the first place. Unfortunately, even after class, I still became stuck.