Group members: Emmanuel Gil, Christ-Brian Amedjonekou

PHYS 3600ID-D862 (Machine Learning) Morphology of Galaxies Final Project

Date: 5/20/19

***Process Report***

At the beginning of our project Christ and I had to learn what tensor flow and keras were and why we needed them after that. We referred to chapter 13 in our class book, “Hands-on machine learning with Scikit-Learn and TensorFlow”, which covered convolutional neural networks. To install TensorFlow and keras we used the Anaconda Navigator environments tab and searched for uninstalled packages with relation to TensorFlow and installed them to our base(root) environment. We downloaded the full galaxy data set from Kaggle and looked at all the files and folders within it.

As the topic was new to Christ and me we had to complete and evaluate examples from the class book that taught us what filters and convolutional layers were and how to create filters and convolutional layers to apply to images and output new images. We chose to create a convolutional neural network because, it seemed to be a great choice for image data generation and extracting features from images. We began to try to apply a convolutional neural network on a dataset of hand symbols of rock, paper and scissor by reviewing (Laurence Moroney’s & Karmel Allison’s) code. I learned how to use a keras function called ImageDataGenerator which is a function that allows recallability, orientation changes and many more edits to images.

After we set a directory or path to variables for our training and testing sets, we used a function in keras called flow\_from\_directory which takes all the images in the directory and applies the specified target size and class\_mode which in our case since we are predicting between 3 classes we don’t want the default which is set to binary for two classes. After we created the train and test generators we had to create our model to apply our convolutional layers and pooling. ***Talk about how we learned what convolutional layers were and pooling and that we used max pooling 2x2 steps each move, mention the 512 function iterations and 3 neurons or output possibilities. Dense (3, activation = ‘softmax’) is for the number of possibilities or neurons and relu for amount of functions each pixel in the image gets mapped to.***

Talk about model. compile which has the parameters loss and optimizer. The job of the parameter loss is to measure the results of all the times a hand symbol is classified and figures out how bad the classifications were and passes that data to the optimizer parameter which then makes the next guess based on the parameters of the 512 functions in the model and then keeps repeating n times where the parameter epochs = n in the function model.fit\_generator where we also pass in the training and test generators that we previously made. We use model. save to store the features that were extracted and keep the level of accuracy from the end of the last fit\_generator. We then plotted the history and growth of our model and show as it gets better after each iteration. We applied this same method for our galaxy data set but, ran into some problems with creating the training and test generators. We were getting 0 images classified in 0 classes because our data set didn’t have any labels and we are trying to do a semi supervised approach. I analyzed the data and found that the practice rock paper scissor data set had sub folders in the training and test directories where all the images were classified and organized in the respective sub-folder. I concluded that the subfolders were acting as the classes and our galaxy dataset didn’t have any sub-folders. I manually went through the galaxy data set and created 3 sub-folders for 3 classes for our outputs to map to. The folders are elliptical, spiral, and zoutlier (the images that don’t fit the previous two classes). I then classified about 120 images by hand and assigned them to their respective classes for both our training and testing directories. This method fixed our problem and the model ran very well for the limited training images! It got up to 82% accuracy on the test set and 98 for the training. The graph that I plotted plutoed, so a greater n for epochs will only minimize the standard deviation