Homework 2

Kelsey Hawkins

Introduction

ASL is a prominent language in the American deaf community. With this in mind, I have a dataset of images containing the ASL dictionary, excluding the letters that need motion. The dataset was already flattened into greyscale values, ready for model input with a few preprocessing changes. This model will help translate ASL images to text for those who do not understand it, or want to learn it.

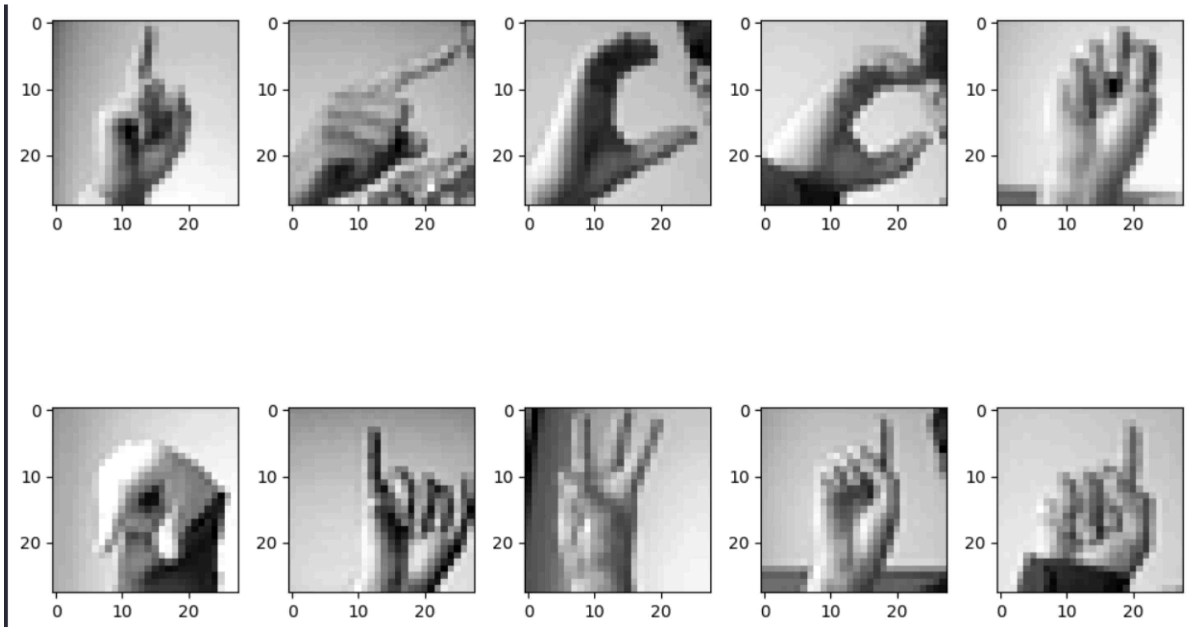


Figure 1: Random Viz Images

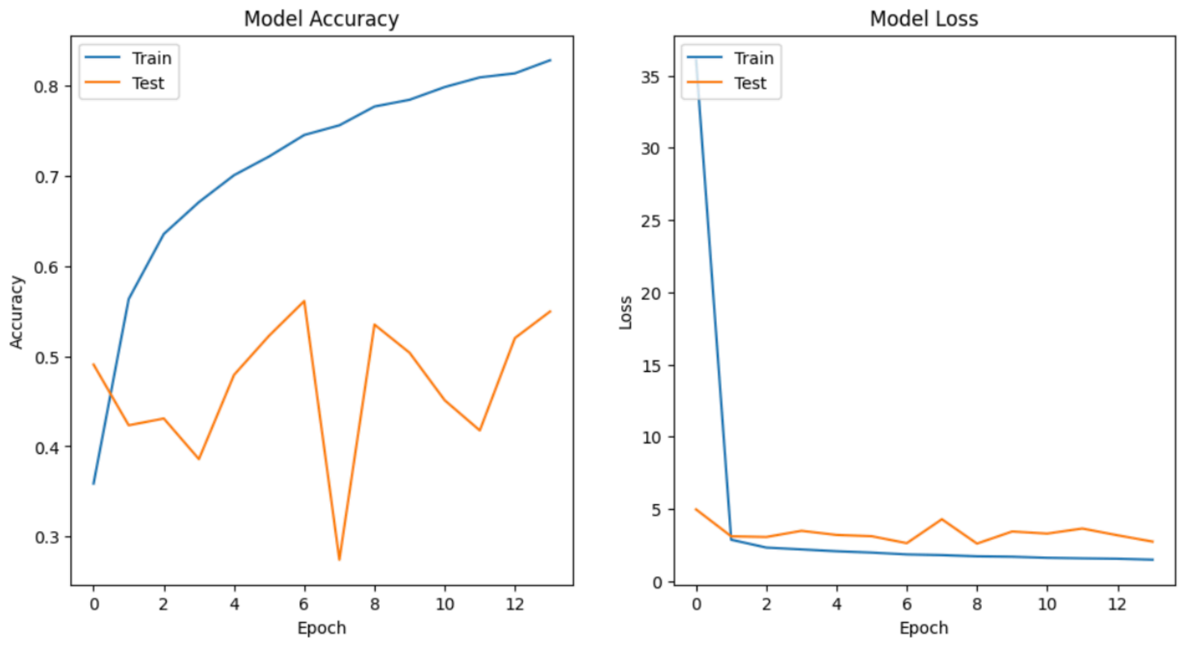
Analysis

The first exploratory item I did was visualizing a few images from the dataset. Not only did I want to ensure the data accuracy, but I wanted to see what resolution I was working with. It was hard to do any summary statistics or heatmaps, etc. due to the data being flattened images so no EDA would have been useful in this case.

However, the images have a 28 x 28 x 1 size, a very low resolution image of hands depicting each letter of the american alphabet. Since the images have been flattened into columns of greyscale values, there will be 784 columns, and all values except for the predictor variable were scaled between 0-1. The predictor variable is a single number that denotes a letter of the alphabet, excluding J and Z which require motion.

Methods

The first model I created is a Deep Feed-Forward Neural Network. I started with a simple model, built in class that had about 6 hidden layers. I then expanded it by adding more dropout, dense, and batch normalization layers. Each dense layer has an activation function of reLU, and a kernel regularizer of L2 on two early dense layers. The last dense layer outputs 24 nodes with a softmax activation function for predicted probabilities for each letter. When I compiled my model, I tested multiple optimizers including SGD, Adam, and RMSprop. The model included early stopping to prevent overfitting.



{width = 80%}

The second model I created is a Random Forest Classifier. I created a simple one to get a general sense of the model outputs and comparison to the DNN. Then, I built a tuned RFC by utilizing the Randomized Search CV module. The parameters I tuned were the number of estimators, max depth, min samples split and leaf. The scorers were a combination of accuracy and F1 for a combination between all three classification metrics.

Results

With my two models, it is proven that deep learning is not needed. In the deep neural network, no matter how many layers I added or removed, or tuning I did, the metrics did not significantly improve. Both the accuracy and F1 scores were extremely low and the model failed to fit to the data well. Especially since I could not use convolutional layers, the neural network failed to find features well. The training and testing scores were not far from eachother, so no underfitting or overfitting was present. However, the testing accuracy was 34%, and the F1 score was also 34%. Paired with the plot below, this shows the model does not find meaningful features and patterns with the data, and it should not be utilized in a scaled environment.

In the random forest classifier model, the metrics were quite different than the DNN. This further proves deep learning is not necessary in this context as the comparison of scores were multitudes different. The random forest classifier outputted very high metrics once it was tuned. The tuner found the best number of estimators to be 300, the min samples split of 2, and min samples leaf to be 1 and max depth of 20. The training and testing accuracy showed signs of overfitting as the training accuracy was 100% and the testing accuracy was 83%. In the images below, we can see the predictions were valid for all five images, as opposed to the predictions made by the DNN.

Reflection

In this assignment, I did not come across many struggles or problems with the code or analysis. It was very interesting to look into the architecture and math behind the models to prove why the neural network did horribly compared to the random forest classifier. I think this assignment was great to prove that you don't always need deep learning architectures with your models, and sometimes simple models work best.