Final MINI Project

Mehrab Mehdi Islam
mehdiisl@ualberta.ca
1545664

Introduction & Description

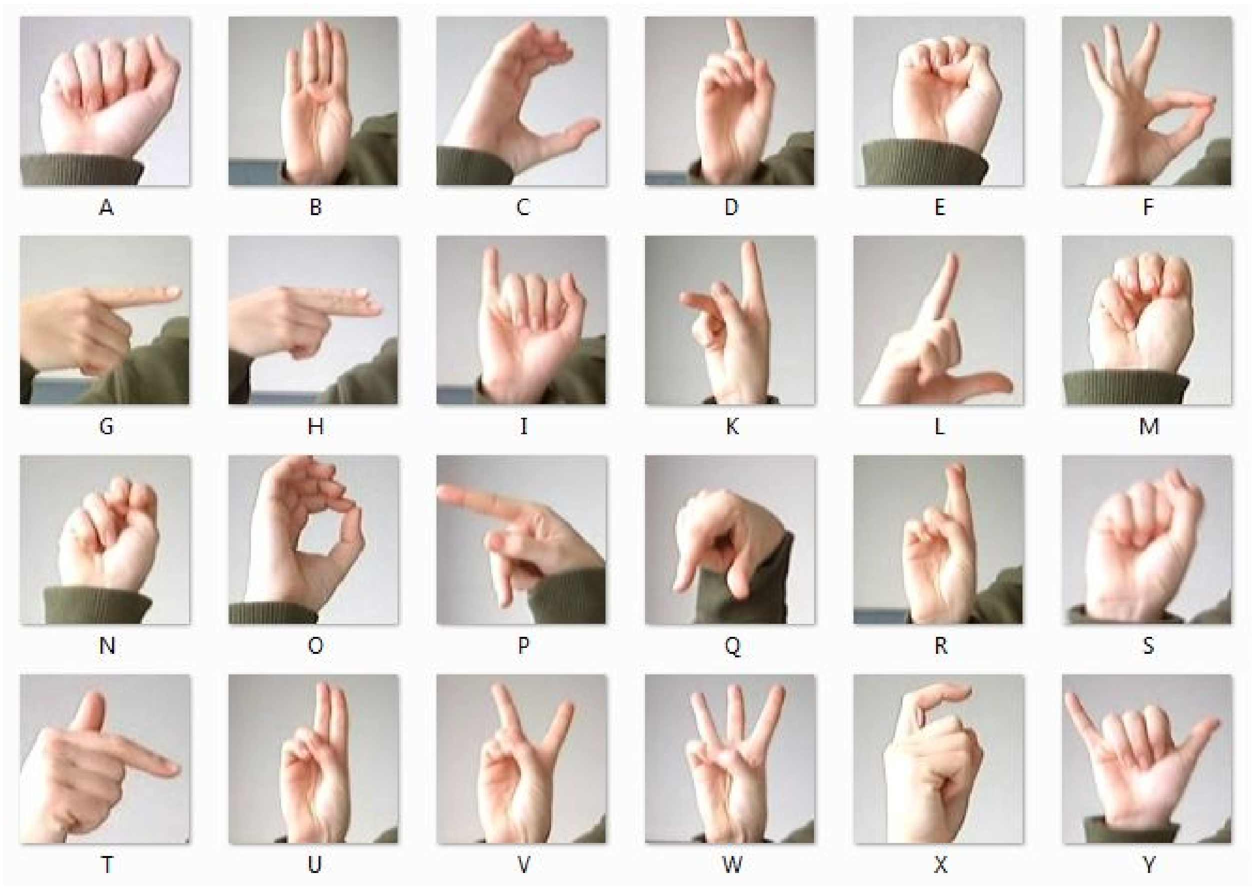
This project is based on American Sign Language letter database of hand gestures .The dataset follows the CSV format of having labels and pixel values in single rows. The American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion)

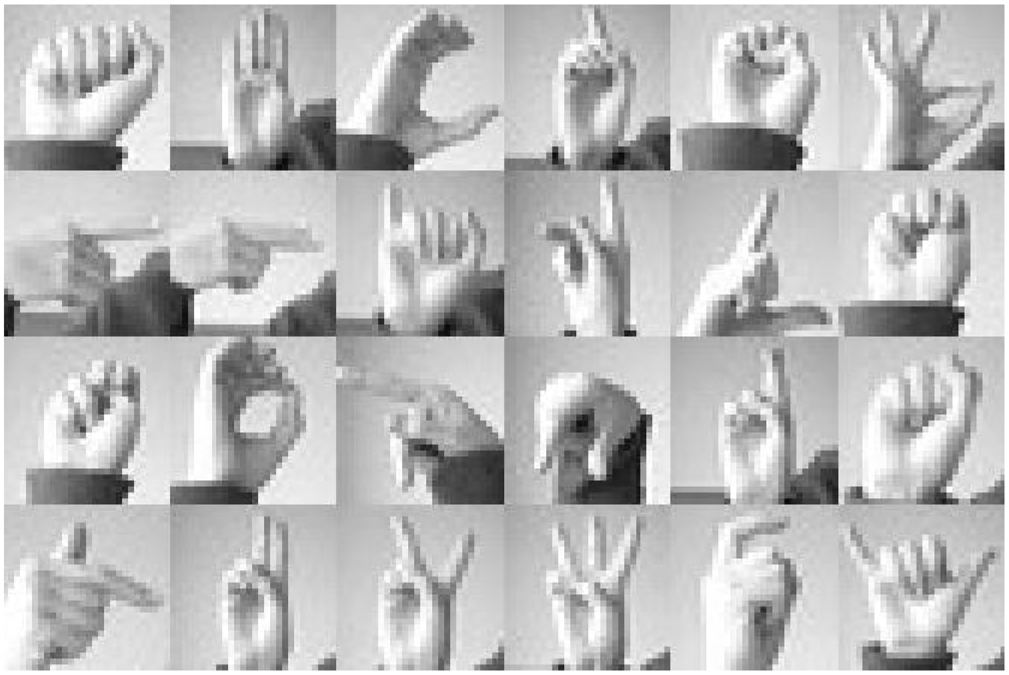
The data consists of the test set and train set which are samples of handwritten digit 28x28 pixels images. The values are numerical ranging from 0 to 255. The target variable is the class of the image - which is the letter (in digit) it contains.

The train set has 27,455 cases with 784 features and 1 target . The test set has 7172 cases. Each sample point has 785 columns. The first column, called "label", is the letter. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixelx, where x is an integer between 1 and 784, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j +1, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions. The picture below shows the ASL image with the corresponding letter. With 0 mapped to A and incrementing from there.





The black and white picture shows example of the greyscale ASL signs from the dataset

Objective

In short, the dataset can be used to train an ASL language Recognizer for NLP's . AI systems such as Google Assistant or Amazon's Alexa can have a built in camera in the future so that people who aren't able to speak can still execute commands on these devices as long as they are in range of the camera attached to them.

Implementation

This project will be using a training-validation-test infrastructure to get our results.The train dataset will be split into a Train and Validation dataset with 24010 training samples and 3421 Validation samples.The model will be trained on the train dataset and then predict onto the Validation and test datasets.

This project will use the 3 following Machine Learning algorithm:

- K-nearest neighbors: KNN classifier is a simple algorithm that does not require learning. The algorithm calculates distances between feature vectors of the test sample and training set. It then selects the K nearest training data point. The test data point belongs to the class that appears the most in these K points.
    - Implementation: Use KNeighborsClassifier from sklearn.neighbors, fit classifier with data from training set, predict label for test set.Training time is measured as execution time of "fit(Xtrain, ytrain)" function. Predicting time is measured as execution time of the "predict(Xtest)" function.
    - Hyper Parameters: K =[2,3,4,5,6,7,8]
    - Where the Hyper Parameter K is tuned with the values in the array
- Logistic Regression: Using stochastics gradient descent solver to learn weight vector from training set.
    - Implementation: Use LogisticRegression from sklearn.linear_model, fit with training set, predict label on test set. Training time is measured as

execution time of the “fit(Xtrain, ytrain)” function. Predicting time is measured as execution time of the “predict(Xtest)” function.
    - Hyper Parameters:C = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 , 0.9], random_state=0, solver='sag', multi_class='multinomial', max_iter=100
    - Where the Hyper Parameter C is tuned
- Neural network - 2 hidden layers with 100 nodes in each.
    - Implementation: Use MLPClassifier from sklearn.neural_network. The data will be normalized by dividing the features from the test set and train set by 255 before training. Training time is measured as execution time of the “fit(Xtrain, ytrain)” function. Predicting time is measured as execution time of the “predict(Xtest)” function.
    - Hyper Parameters: hidden_layer_sizes=[(50,50,50), (50,100,50), (100,), (100,100)], alpha = [0.0001, 0.001, 0.01, 0.1, 0.9, 1], max_iter=400, alpha=1e-4, solver='sgd', tol=1e-4, random_state=1
    - Where the Hyper parameters alpha and hidden layer sizes are tuned

Results

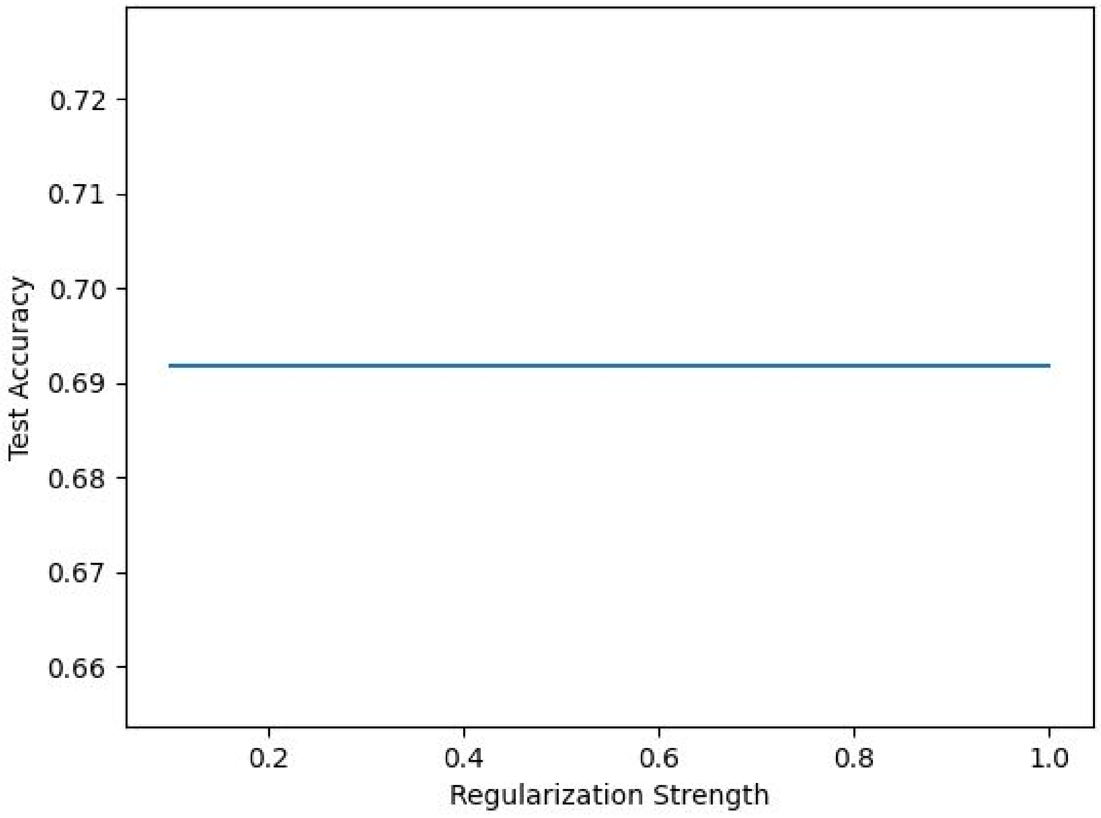
Logistic Regression:

Logistic Regression with different Regularization strengths-
Train Times = [279.32396125793457, 279.6940658092499, 279.3723485469818, 280.01038694381714, 279.59876894950867, 270.3488664627075, 269.9667329788208, 270.0525040626526, 270.717232465744, 152.3005495071411]

Predict Time = [0.03314375877380371, 0.030011892318725586, 0.03133130073547363, 0.022151708602905273, 0.05192804336547851 6, 0.023096084594726562, 0.043111562728881836, 0.023284912109375, 0.023902416229248047, 0.017948389053344727]

Accuracy = [0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043, 0.691717791411043]

Best Accuracy = 0.691717791411043



K-nearest neighbors:

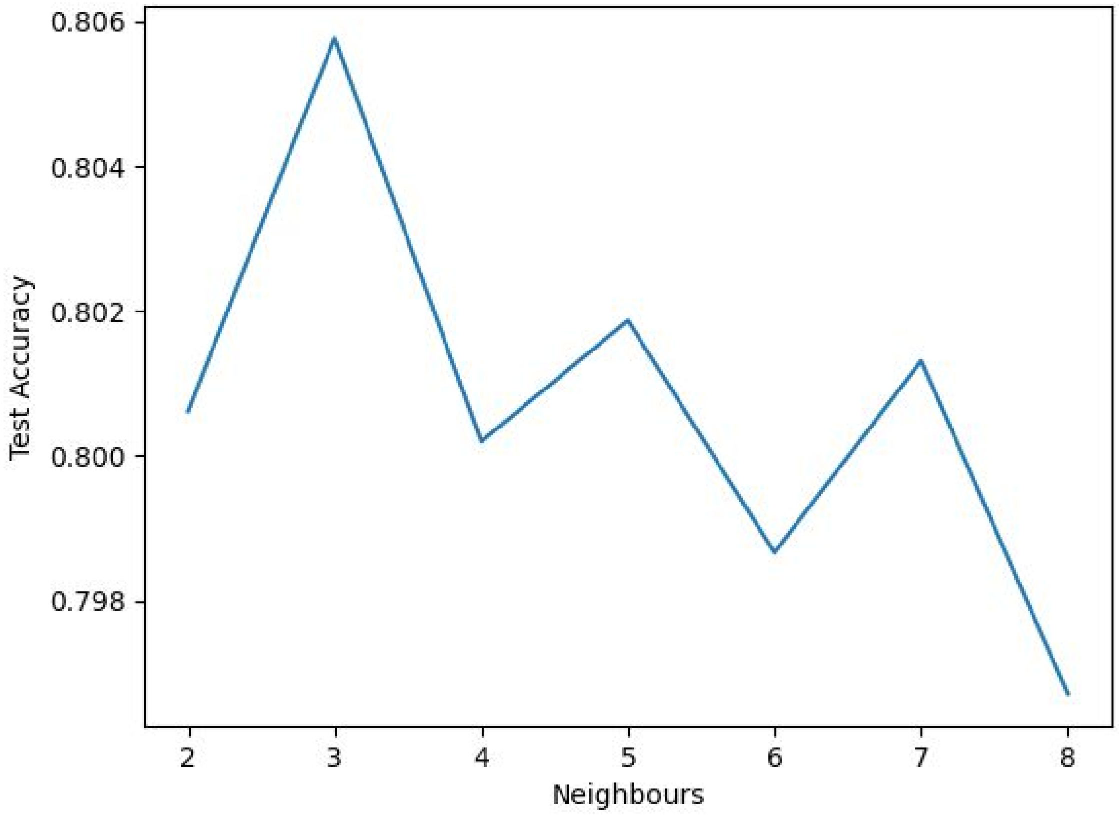
K-nearest neighbours with different number of neighbours

Train Times = [3.4072751998901367, 3.4000468254089355, 3.338242530822754, 3.2803080081939697, 3.3285868167877197, 3.2627851963043213, 3.372084856033325]

Predict Time = [248.25368452072144, 254.0865340232849, 262.88110160827637, 264.939581155777, 269.0768213272095, 273.75235056877136, 279.1719717979431]

Accuracy = [0.8006134969325154, 0.8057724484104852, 0.8001952035694366, 0.8018683770217513, 0.7986614612381484, 0.8013106525376464, 0.7967094255437814]

Best Accuracy = 0.8057724484104852



Neural network:

Using different values of Alpha (l2 regularization penalty), we plot 4 different graphs based on the hidden layers.

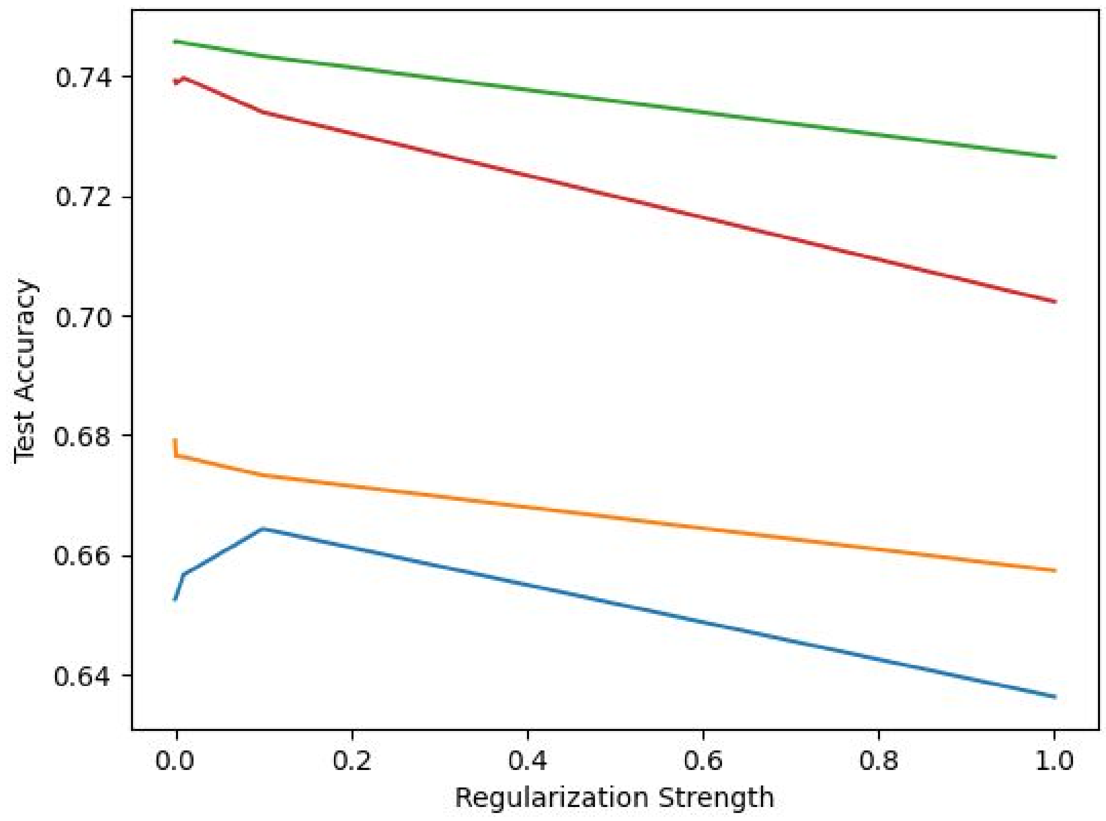
(Best result - green Data)

Train Times = [269.7104411125183, 207.85859155654907, 212.27260851860046, 270.5550413131714, 273.8849866390228]

Predict Time = [0.05808091163635254, 0.06078982353210449, 0.05725169181823730, 0.05221080780029297, 0.07108116149902344]

Accuracy = [0.7392638036809815, 0.7387060791968767, 0.7396820970440603, 0.7339654210819855, 0.7023145566090352]

Best Accuracy = 0.7396820970440603



Blue:-(50,50,50), Yellow - (50,100,50), Red:- (100,), Green- (100,100)

<u>Analysis</u>

Our data shows that KNN offers the best Results if we look just directly at the accuracy of the tests. They give the best result with 3 neighbours and have a very miniscule change in accuracy when the number of Neighbours are changed. In terms of Accuracy Next is the Neural Network Approach which gives very different and varying results based on the finite set of hyper parameters it was tested on as shown in the results. In its best state using (100,100) hidden layers, it got an accuracy of about 74% with a very miniscule change due to the regularization Strength. Logistic Regression gave by far the worst results in terms of Accuracy as it had an accuracy of 69% with nearly no change when you tune the hyper parameters. In terms of Speed, Both Neural Networks and

Logistic Regression had slow training times (200-300 secs) and fast prediction times (Less than 0.1 sec), Whereas K-nearest neighbors have low training times (less than 5 secs) but high prediction time(200-300 secs).

Conclusion

The Optimal situation for this problem would be having an high accuracy with low prediction time as the prediction will be done in real time whereas the training for the ASL can be done before hand would not need much change at all as the signs remain constant

Based on this I believe that a NN approach would be better for classifications tasks running in real time. While KNN would be ideal for tasks that do not require real time application but requires classification on new and/or varying types of data often. Logistic Regression got similar type but poorer results than NN, so it does not need to be taken into account as we can just use KNN for classification tasks like this as this would just give a better result overall than Logistic Regression.

For this specific task, to reach the objective that i stated above, I think it would be best to use NN. This is largely due to the fact that AI tools such as Google Assistant or Alexa need to respond as fast as possible, hence we would NN as it predicts much faster, while trying to improve the accuracy of the NN by using more samples in the training while trying to prevent the bias of identifying just the training type images

Sources:-

Database:

1. https://www.kaggle.com/datamunge/sign-language-mnist