Project 2: Human Observed Dataset and Gradient Structural Concavity Algorithm Dataset Handwritten Character Prediction

Introduction to Machine Learning

Yash Narendra Saraf
UB Person No. 50290453
MS Computer Science and Engineering
School of Engineering and Applied Science
ysaraf@buffalo.edu

Abstract

In this project we have two types of features collected from the same dataset. Several writers were asked to write a particular word "and" , this word was used to recognize the pattern in the written character to differentiate between different writers. The features were extracted by two methods i.e Human Observed Dataset and the Gradient Structural Concativity. So here we are trying to match the two instances of word dataset and then try to predict that if the two instances have been written by the same writer or different writers.

1 Datasets

The data could not be uploaded to the Timberlake so here is the drive link for the entire dataset.

Google Drive Link for the the complete data set. Click on this hyperlink

1.1 Human Observed Dataset

The human observed dataset has been created manually by observing the patterns in the datset and assigning a number between 0 and 4 for 9 such features for each instance of the word "and". The number of datapoints we have for the Human observed dataset are low for obvious reasons. So here we first create the same pairs i.e image ID's by same writer which is mapped to target 1 i.e. perfect match and the different pairs i.e. image ID's by different writer's which is mapped to target 0. So for the number of different pairs is going to increase expontially whereas the number of the same pairs will increase as O(1) so to balance the datset what we do is pick limited number of different pairs which is closely equal to the number of same pairs.

1.2 Gradient Structural Concativity

GSC is a complex automatic way to extract features from the images. The Gradient and Structural features encode the local structure, while the concavity features are global descriptors extracted from the binary images. So we have 512 features for each image and each of the binary features describes the pattern in the image. So this is calculated for all the dataset. The here also we create the same and different pairs and use the same for feature recognition.

2 Pre-Processing

2.1 Seen, Unseen, shuffled pairs

We have divided the dataset using three statergies. These 3 statergies has been explained below.

The seen writer means that each writer is present in all the three datasets i.e. training, testing, and validation datset. This is done to ensure that when we are testing it on the new dataset we have at least seen that writer once. So according to intuition this type of data division should have higher accuracy as the writer is already been processed. Make sure that we have only the same writer not the same exact pair. Having same exact pair is not training as we are then testing and training on the same data.

The unseen data partitioning represents partioning the data into the three parts such the the three data sets have no common writer. This type of partioning is more close to real world scenario as when we partition the data we have the assurance that the same datapoint has not been seen before. This type of model might not get trained to the same accuracy as the seen writer dataset according to intuition.

The shuffled random dataset is essentially randomly picking up the datapoints from same and different pairs. So here we have some overlap between all the 3 datasets when we talk about writer's. This kind of partioning is quite simple and does not require much processing

2.2 Subtraction and Concatenation

Now since we need to compare two pairs to data we need to combine two instances of features. Now this can be done in many ways, but the two ways we are considering over here are processing the data by subtracting corresponding features as it makes sense to have the difference low when we have the same writer instance and another method adopted is concatenation, here the loss of information is nill. The concatenation according to intuition must perform better as the correlation between the features would be better mapped by high dimensional neural network.

3 Theory - Methods Used

According to the project description we are expected to work with three types of models i.e. linear regression, logistic regression, and Neural Networks. There three are the fundamentals of Machine Learning and have been experimented with thoroughly throughout the project.

3.1 Linear Regression

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. Its used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog).

3.1.1 Radial Basis Function

In multidimensional regression X = RD. The easiest way to attack the regression problem is to look for f in a finite dimensional space of functions spanned by a given basis. In other words, we specify a set of functions 0, 1, . . . , P from X to R and look for f in the form of a linear combination.

We can have multiple types of Basis functions like the simple linear regression, polynomial regression and the Gaussian RBF'S. Here we have used the Gaussian RBF's. The formula for the same is as follows -

$$\phi = e^{-1/2*(x-\mu)^T*\Sigma^{-1}*(x-\mu)}$$

Here μ denotes the mean matrix, x denotes the input matrix and the Σ denotes the covariance matrix.

3.1.2 Closed Form Solution

We know that the target can be evaluated using the equation.

$$t = w^t * \phi(x)$$

So to find out the weights vector it is necessary to findout the inverse of the design matrix. Since the design matrix has very high dimensionality and also the matrix is not a square matrix. We need to use the Moore Penrose Inversion i.e. the pseudoinversion of the matrix which gives generalization of the inversion matrix. So to make sure we can find inversion we need to make sure that none of the columns has zero variance. The zeros variance is an important task as the matrix is not invertible even if one of the rows or columns has zero variance. Using the following moore penrose formula we can obtain the weight matrix. This is then used to calculate the output of for all the three sets i.e. validation, testing and training. Here it does not make sense to have validation split seperately. Still a validation split has been maintained so as it is being used in the Stochastic Gradient Solution.

$$w = (\phi^T \phi)^{-1} \phi^T t$$

3.1.3 Stochastic Gradient Descent

This is a linear regression solution uses the Stochastic Gradient method to update the weights after each iteration. Over here the output is calculated based on some random predetermined weights, these weights are then being updated after every iteration. Here the output is being compared to the target value and and RMS value is calculated. To reduce the error value the gradient of the error is calculated and based on some constant factor the weights are then changed. The constant factor is known as the learning rate. This is a hyperparameter and should be adjusted carefully. The stochastic gradient equations are explained below.

$$w_i = w_i + \eta \Delta w_i$$
$$\delta E / \delta w_i = \delta / \delta w_i [1/2 * (target - output)^2]$$
$$\delta E / \delta w_i = -\phi(x_i)[target - w^T \phi(x)]$$

3.1.4 Regularization

The model when trained just by using gradient descent may start to overfit the testing data. To avoid overfitting various techniques are used i.e. using regulariaztion with weights and using the validation data split. The regularization term is added to the error and the sum is then reduced which makes sure that the weights also stay within the bounds and avoids overfitting. Validation is usually used for early stopping where we check the validation set after every few iterations of the training set. As the validation accuracy starts to decrease for few significant iterations, it means that the model has started to overfit the training set and the computations need to be stopped.

$$E_D(w) = 1/2 \sum_{n=1}^{N} (target - w^T \phi(x_n))^2$$

The above equation denotes the total error term.

$$E(w) = E_D(w) + \lambda E_W(w)$$
$$E_w(w) = 1/2 * w^T w$$

The regularization term has been added to the Error term to make sure the model does not overfit and the weights do not go out of bound. The regularization also keeps the weights to be minimum.

The closed form solution with regularizer looks like-

$$w^* = (\lambda I + \phi^T \phi)^{-1} \phi^T t$$

3.2 Logistic Regression

The logistic regression is a classification technique. This is used when the given target value is Binary. For our problem this seems to be a better model than the linear regression as we are trying to classify into two buckets. The difference between the linear regression and logistic regression is quiet small. Here we use Cross Entropy as the loss function and calculate the output using sigmoid activation function, for a multiclass problem the output is calculated using the softmax function which is nothing but an extension of the sigmoid function. The model connections are all the same. What logistic regression essentially does is try to find a hyperplane which seperates the data into multiple classes.

$$prediction = sigmoid(W^T X)$$
$$prediction = \frac{1}{1 + e^{-W^T X}}$$

The above equation is just applying the sigmoid function to the simple output of the linear regression. The main function of the loss function is to get the idea of how far are we from the actual target and this job is better done by the Cross Entropy loss function which will be shown below. Also the Mean square error is non convex when it comes to logistic regression problem and this might cause us to converge at a point that is not the required minima. Also using the Cross Entropy provides us with the same gradient function which is shown below.

$$CostFunction = \sum_{n}^{alldatapoints} -((1 - target) * log(1 - prediction) + target * log(prediction))$$

Now to minimize the cost function we will differentiate the cost function with respect to weights. So that we can use Stochastic Gradient Descent to update the weights to the optimal values.

$$\frac{\partial CostFunction}{\partial W_j} = -\left(\frac{target}{sigmoid(W^T X)} - \frac{1 - target}{1 - sigmoid(W^T X)}\right)\frac{\partial sigmoid(W^T X)}{\partial W_j}$$

$$= -\left(\frac{target}{sigmoid(W^T X)} - \frac{1 - target}{1 - sigmoid(W^T X)}\right) sigmoid(W^T X)(1 - sigmoid(W^T X))\frac{\partial W^T X}{\partial W_j}$$

$$= -\Big(target * (1 - sigmoid(W^T X)) - \big(1 - target\big)\big(sigmoid(W^T X)\big)\Big) X_j$$

$$= -(target - prediction) X_j$$

Here we can see that the gradient equation remains the same as the linear regression. Now we use the same L2 regularizer term and the equation for the gradient now is

$$Gradient = -(target - prediction) X_j + \lambda * \sum_{n}^{allWeights} W_j$$

Using the above gradient equation we update all the weights.

To optimally use the GPU, I have also written a keras code for linear and logistic regression. So we have used only two layers i.e. input layer which passes all the inputs to the second layer. Now the second layer has only one neuron.

Now if we use the mean square error as the error function and the Stochastic gradient descent as the optimizer we have succesfully modelled the linear regression problem.

Now if we also apply the sigmoid activation to the output neuron and also change the loss function to cross entropy we have successfully modelled the logistic regression in keras.

The adjacent figure shows the corresponding model which has been extracted from the tensorboard.

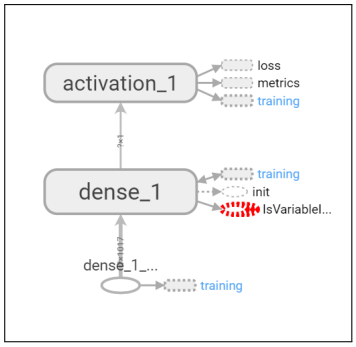

Figure 1: Model for Logistic/ Linear Regression

3.3 Neural Networks

Regression is method dealing with linear dependencies, neural networks can deal with nonlinearities. So if your data will have some nonlinear dependencies, neural networks should perform better than regression. Linear regression can be thought as the 2 layer neural network which connects all the inputs to a single neuron in the output layer which has no activation function, and the error is calculated using the mean square error. Now since we have linear equation we can only draw a hyperplane to map the input features to the output features.

But the neural networks are capable of a lot more, i.e. they can successfully map non linearities present in the data as the linear combinations of inputs are first passed to the hidden layer which is passed to a suitable activation function, now this represents nothing but a set of new features which are formed by the linear combination of orignal features. So this helps in mapping non-linearities in the neural networks.

Furthur the model of the neural networks used and the parameters associated would be explained here.

I have used the keras model where we have used a sequential model i.e. every layer connected to the next one via weights. Here a particular example for all parameters has been used to explain the model.

Consider the human observed dataset where we have 9 features corresponding to each writer and then we we create user pairs then we concatenate the features giving us a total of 18 features which is the number of neurons in the input layer.

Then the activation function has been applied, in our case a relu function so that the range of values can be constrained to a particular limit.

Then to avoid overfitting a dropout layer has been added so a few neuron outputs are dropped to 0 randomly before transmitting to furthur layers.

To map the complexity of the given dataset we have used two hidden layers i.e. the second layer also has a corresponding activation layer along with it. Same relu function has been used again.

Then a dense layer 2 has been added which has 2 nodes i.e. the number of classes. The number of weights associated is shown in Table 1.

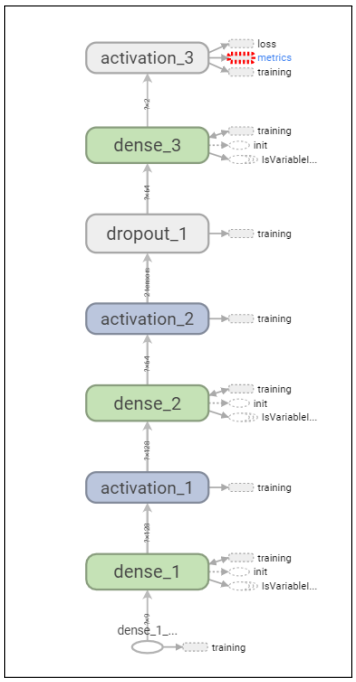

Figure 2: Model for Neural Network

The sigmoid activation is applied to find the probablity of the input being in each class to make the final guess.

Layer	Number of nodes	Weights associated or application
Input Layer	18 nodes	None
dense_1 (Dense)	128 nodes	(18+1)*128=2432
activation_1 (Activation)	128 nodes	Relu applied to all dense_1 outputs
dropout_1 (Dropout)	128 nodes	Some percentage of nodes nullified to 0
dense_2 (Dense)	64 nodes	(128+1)*(64)=8256
activation_2 (Activation)	64 nodes	Relu applied to all dense_2 outputs
dense_3 (Dense)	2 nodes	(64+1)*2=130
activation_3 (Activation)	2 nodes	sigmoid applied to find probablity

Table 1: Model Description for Neural Networks with respect to Human Observed dataset Concatenation of features, The conditions considered over here are we have 2 hidden layers and relu has been applied to both of the hidden layers and also we use two output neurons for the output buckets.

4 Experimentations and Observations

4.1 Linear Regression

Linear Regression has been implemented in 2 ways, i.e. without using complex libraries like tensorflow and keras and the other implementation has been done by using keras. All the processing for the keras code has been performed on GTX 1060 GPU. The linear regression has been performed on all the datasets we have present. The implementation has been explained before where we use basis function to reduce the feature space and train the network using the new basis features. Here we have tabulated he results for the performance of the linear regression model on all the datasets we have present.

Now since the Human observed dataset lacks number of features and also there might be human error associated with the dataset the training on human observed dataset has been poor and gives an accuracy in the range of about 50-60 percent as can be seen from the graph. Here there are no significant differences for the network in performance on different datasets.

Partition	Sub/Concat	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.54469	52.68542	0.53809	48.75622	0.521907	47.12644
seen	concat	0.4988455	51.74766	0.495113	55.22388	0.460829	83.90805
unseen	Sub	0.49815	55.97738	0.51421	51.47059	0.513881	48.90282
unseen	concat	0.49534	55.41195	0.49952	52.94118	0.508272	49.2163
shuffled	Sub	0.50359	54.64521	0.50995	52.04678	0.516107	55.55556
shuffled	concat	0.69123	53.32846	0.66846	54.38596	0.697278	54.38596

Table 2: Human Observed Dataset Results for Linear Regression

For the GSC dataset since we have sufficient number of features the performance has significant improvement when compared to the Human Observed Dataset. Also the basis function has been used only for a few datasets as calculating the centers using the Kmeans clustering, and generating the Phi matrix is a very slow process for such a large dataset. Now while working with the GSC dataset we had to be carefull as the Phi values were large when we calculated them using provided formulas. So this was causing the model to drift apart when it came to predictions and then later give a math error as the predictions overshoot out of bounds. So to avoid this we upscaled the BigSigma matrix by 2000 which again brought back the predictions in proper range.For the GSC we have tabulated the resuls below. For the below predictions no basis functions has been used as the use of basis functions was taking up a lot of time for the creation of Phi matrix.

Partition	Merge	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.2483	0.5405	0.2514	0.5161	0.255863	0.489046
seen	concat	0.2225	0.6523	0.2225	0.6378	0.205362	0.678759
unseen	Sub	0.2483	0.5401	0.2527	0.5217	0.25642	0.501979
unseen	concat	0.2156	0.6678	0.2629	0.5892	0.31624	0.517878
shuffled	Sub	0.248	0.5424	0.2494	0.5379	0.250041	0.533381
shuffled	concat	0.2221	0.6536	0.2378	0.612	0.232929	0.622483

Table 3: Linear Regression for GSC dataset

4.2 Logistic Regression

Logistic regression has also been performed by using both keras and non complex library implementation. Here for the human observed dataset there are not significant changes in accuracy, i.e. the accuracy range lies between 50-60.

Partition	Merge	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.7191207	51.83291	0.699353	53.23383	0.594813	73.56322
seen	concat	0.7264544	50.98039	0.693022	55.22388	0.529676	86.2069
unseen	Sub	0.68544	54.20032	0.69148	46.32353	0.709136	48.90282
unseen	concat	0.67916	56.21971	0.68711	52.20588	0.723292	49.2163
shuffled	Sub	0.69051	50.40234	0.67101	56.14035	0.698168	54.97076
shuffled	concat	0.68827	53.69422	0.67011	56.72515	0.694971	55.55556

Table 4: Human Observed Dataset Results for Linear Regression

But for the GSC dataset the logistic regression does'nt work as well as we expect it to. But still it can be seen that the logistic regression provides with better results than the linear regression which is because we have a dataset which is more suitable for the classification.

Partition	Merge	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.6863	0.5522	0.6945	0.5271	0.704248	0.493888
seen	concat	0.5992	0.6815	0.5871	0.692	0.587408	0.685347
unseen	Sub	0.6868	0.5522	0.6927	0.5243	0.700576	0.501107
unseen	concat	0.5942	0.6858	0.6959	0.5725	0.749097	0.537667
shuffled	Sub	0.6858	0.5543	0.6896	0.545	0.690615	0.538496
shuffled	concat	0.5883	0.691	0.6006	0.6805	0.591312	0.683093

Table 5: Logistic Regression for GSC dataset

4.3 Neural Networks

Neural Networks can be considered as an extension of the logistic regression, but the advantages of neural network are that it can figure out the non-linearities on its own given an appropriate size model. So we use neural networks now to see if the model is able to classify the given into two buckets or not. So we created a four layer model and varied the number of neurons in both the hidden layers and observed the respective changes.

Again since we don't have sufficient features in the human observed dataset we were not able to map even using the neural networks. But for the GSC dataset neural networks worked like wonders and provided an accuracy of more than 98 percent consistantly. Now we also tested the model on various datasets and the performances has been tabulated and explained below.

Partition	Merge	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.3965	0.8363	0.8822	0.4826	0.896412	0.586207
seen	concat	0.6861	0.5533	0.6858	0.5771	0.682167	0.586207
unseen	Sub	0.3565	0.8506	0.9242	0.5074	0.902486	0.523511
unseen	concat	0.003	1	4.221	0.5441	4.927634	0.520376
shuffled	Sub	0.4018	0.8383	0.8448	0.5088	0.830004	0.555556
shuffled	concat	0.4979	0.7696	0.8475	0.5029	0.733988	0.625731

Table 6: Human Observed Dataset with Neural Networks

The performance of the GSC dataset for the Neural network model has been tabulated below. Since the other models were not successfull in predicting for any data sets it was difficult to differentiate between the performance when it came to the performance based on each dataset. Here we can clearly see that the models performance on all three datasets on two different observations.

Partition	Merge	Training Cost	Training Accuracy	Validation Cost	Validation Accuracy	Testing Cost	Testing Accuracy
seen	Sub	0.0124	0.9962	0.0881	0.9812	0.089602	0.981108
seen	concat	0.0226	0.9954	0.0573	0.9857	0.059406	0.986188
unseen	Sub	0.0239	0.995	4.1093	0.5808	3.45454	0.649628
unseen	concat	0.0319	0.9926	4.4642	3.740565	3.740565	0.623063
shuffled	Sub	0.0084	0.9975	0.1051	0.9795	0.117202	0.97902
shuffled	concat	0.0208	0.9954	0.0534	0.9881	0.072032	0.98556

Table 7: Neural Networks performance for GSC dataset

The neural network performs best for the seen dataset which is expected as the neural network has seen the similar pattern before and can easily classify, also for the unseen writer the performance seems to be very bad compared to the other two datasets as the writer has been never seen before the model starts to predict based on the previous seen samples which do not contain the same writer. The shuffled dataset performs just as well as the seen dataset because the shuffled dataset has most of the writers it has seen in the testing and validation whereas the new few writers this model faces get predicted based on the trained model. So the accuracy is slightly less than the seen model.

Following diagrams represent the performance diagrams based on GSC datset for different partioning and merging schemes and also for all three models.

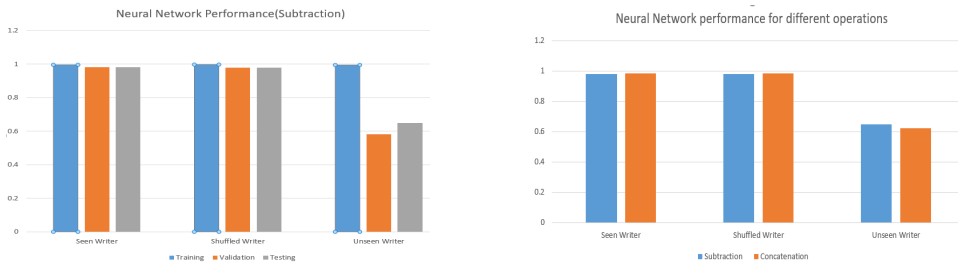

Figure 3: a.Seen ,Shuffled, Unseen Writer comparison for Neural network GSC b. Subtraction and Concatenation comparison for Neural network GSC

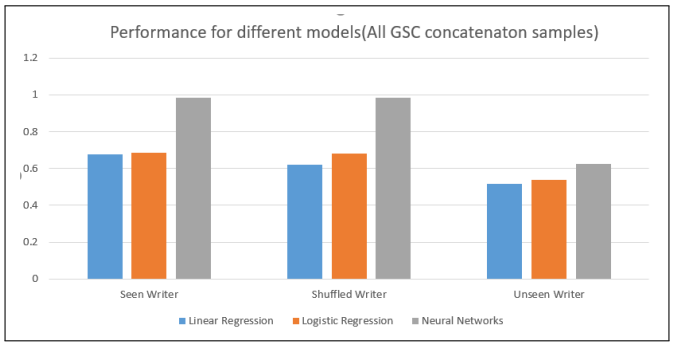

Figure 4: Performance on different models GSC dataset

It can be clearly seen the performance of the Neural Network model exceeds the other two models by a very large margin. Also the concatenation operator gives slightly more accuracy when compared to the subtraction operation and the accuracy when it comes to seen writer is way higher compared to the unseen dataset and also the shuffled dataset performs almost as well as the seen dataset.

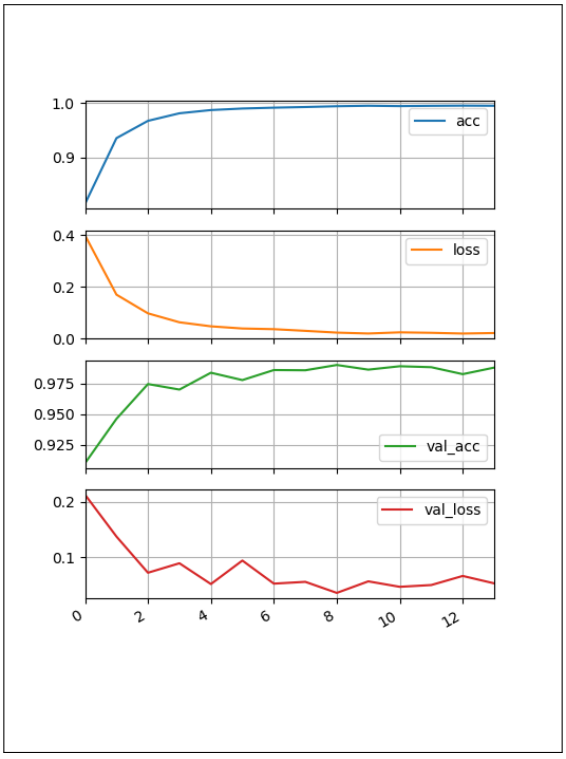

Figure 5: Ideal output which was obtained for the shuffled dataset when features were concatenated. Here we used the 4 layer neural network

The above figure represesnts the one of the most ideal output scenarios where we used thekeras sequential 2 hidden layers to model the complexity of the fiven problem for the GSC datset.

Conclusion

In this project we used two datasets which were extracted from the same set of images using two very different techniques. For the human observed datset we observed that despite of trying different models we are not able to obtain good quality results. This is because the feature set is not sufficient to train the model. The number of features are less and also since it is created manually it has human error associated with it. So these simple models are not sufficient.

But the GSC dataset captures a lot of features for the same sample i.e. 512. So this helps us use simple models to create a model which predicts the output with very good accuracy. With neural networks the model gave accuracies of the order of 98 percent.

Also the three partition schemes and the combination of features play an important role as a part of data processing. The seen dataset performs the best as we would have guessed as the model has seen the same writer before whereas the unseen datset performed poorly. The combination scheme i.e. concatenation gave slightly better results than the subtraction as we can say that the subtraction may also cause loss of data which is not too significant but might cause a small difference which is seen clearly in the results.

References

[1] ResearchGate *Optimizing Binary Feature Vector Similarity Measure using Genetic Algorithm and Handwritten Character Recognition.*
Author : Sargur N. Srihari
https://www.researchgate.net/figure/A-sample-character-and-its-GSC-features_fig1_220861726

[2] Pattern Recognition And Machine Learning, *Book by Christopher Bishop*

[3] Keras Documentation
https://keras.io/