Project 2

Explainable AI: Writer Verification

Anish Gadekar
UB Person No: 50291289
Department of Computer Science
University at Buffalo,
Buffalo, NY 14260
anishraj@buffalo.edu

1. Introduction

The aim of this project is to use a combination of Probabilistic Graphic Models (PGMs) and Deep Learning methodologies in order to develop a Machine Learning system which can provide explainable features. We use a dataset of handcrafted features of the word "and" for this purpose. The dataset is used with Bayesian Networks, Siamese Deep Learning Networks and Auto-Encoders in order to obtain explainable features for the same.

2. Implementation

2.1) Task 1: Generating handcrafted "and" features

In the first task we were asked to annotate 15 set of features for different instances of the word "and". Each student was asked to annotate features for 150 different instances which were combined to create a consolidated dataset of Seen, Unseen and Shuffled datasets which we use for our later models.

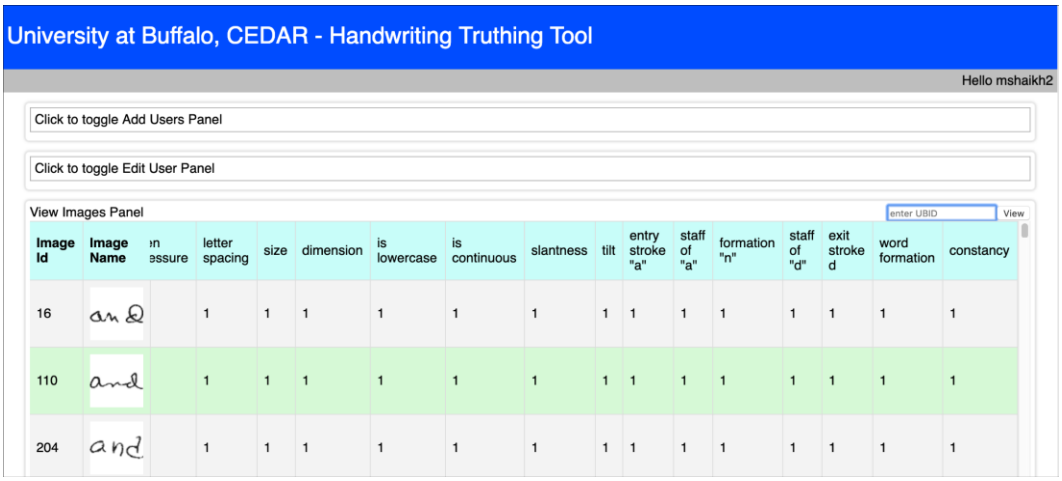

Figure 1: Handwriting Tool

The 15 features are as follows:

1) Pen Pressure: Strong & Medium (2)
2) Letter Spacing: Less, Medium & High (3)
3) Size: Small, Medium & Large (3)
4) Dimension: Low, Medium & High (3)
5) Is Lowercase: No & Yes (2)
6) Is Continuous: No & Yes (2)
7) Slantness: Normal, Slight right, Very right & Left (4)
8) Tilt: Normal & Tilted (2)
9) Entry Stroke: No stroke & Downstroke (2)
10) Staff of "a": No staff, Retraced, Loopy & Tented (4)
11) Formation of "n": No format & Normal (2)
12) Staff of "d": No staff, Retraced & Loopy (3)
13) Exit Stroke: No stroke, Downstroke, Curved Up & Straight Across (4)
14) Word formation: Not well formed & Well formed (2)
15) Constancy: Irregular & Regular (2)

Figure 2: Writer ID: 0968c

Pen Pressure	**Letter Spacing**	**Size**	**Dimension**	**Is Lowercase**
Medium	Medium	Medium	Low	Yes
Is Continuous	**Slantness**	**Tilt**	**Entry Stroke**	**Staff of "a"**
Yes	Very right	Tilted	No stroke	Retraced
Formation of "n"	**Staff of "d"**	**Exit Stroke**	**Word Formation**	**Constancy**
Normal	Loopy	Downstroke	Well formed	Irregular

2.2) Task 2: Bayesian Inference

The aim of this task is to use Probabilistic Graphic Models (PGMs) in order to check whether two "and" instances belong to the same writer.

For the task of Bayesian Inference we used the "15features.csv". The CSV file contained IDs of writers along with the feature values. The writer ID is of the format XXXXY_numZ where XXXX is the writer id, Y is the page number and Z is the occurrence of "and" in page Y. Based on this information, we generated writer pairs with one writer and his respective set of features labelled as left

and the other named as right and on the basis of writer IDs we generated labels for whether the two writer pairs are the same or not called stored in a file called "features.csv" which consists of features of left writer, features of right writer and the target label.

Using the original "15features.csv" we generated a model using the HillClimbSearch available in the pgmpy libraries. The K2 score method was utilized. This model has 42 edges.

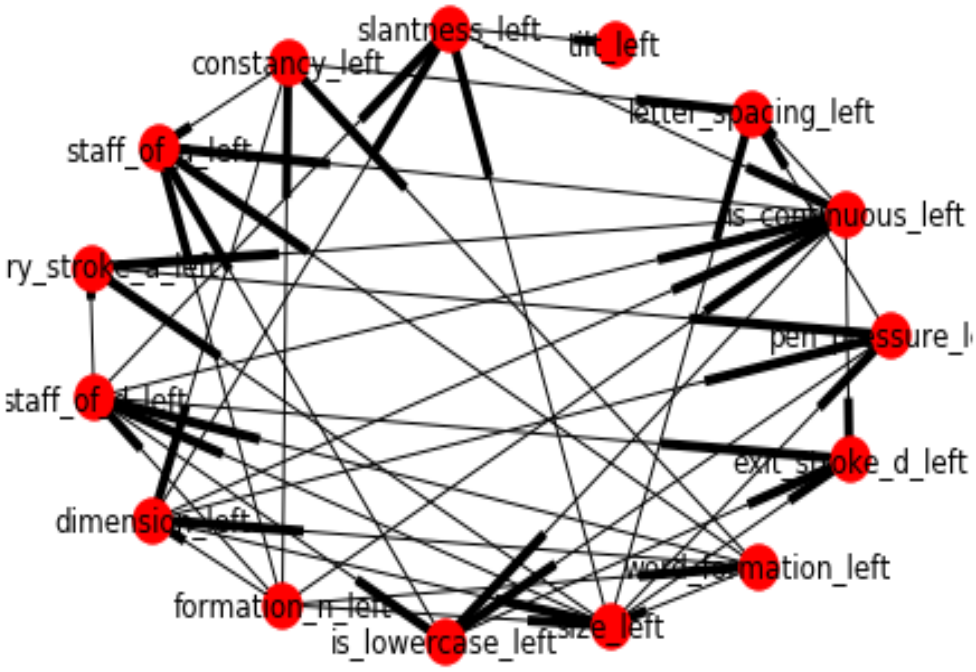

Figure 3: Bayesian Model generated using Hill Climb Search

Using the model generated using the HillClimbSearch method, we generated a clone of the model. The cloned model and the original model were combined to generate the concatenated model which was used with "features.csv". A hypothesis node "h" was added to the model. The nodes "constancy_left", "constancy_right", "letter_spacing_left" and "letter_spacing_right" were attached to the hypothesis node. This model has 88 edges.

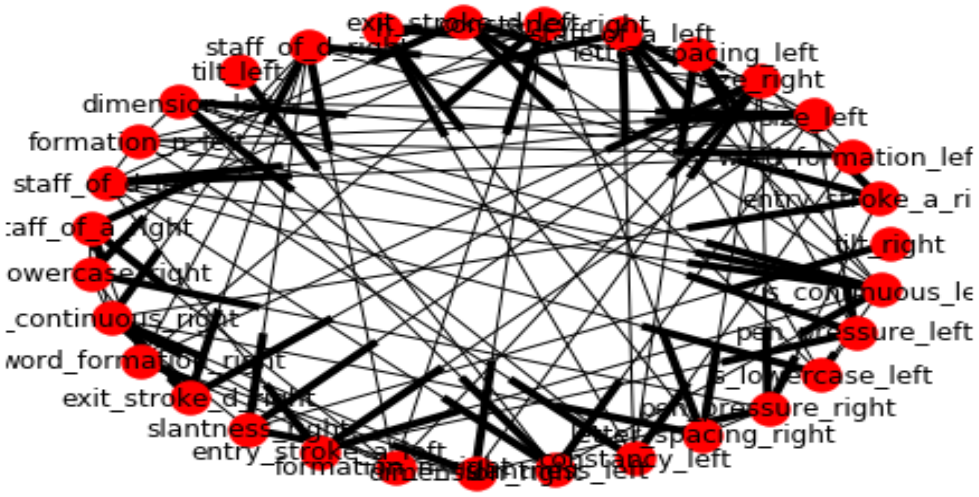

Figure 4: Concatenated Bayesian Network

The generated model was tried on the training and validation concatenated sets. We obtained the inference for the concatenated model using Variable Elimination. A map query was generated for the hypothesis with two cases:

1) With only the nodes attached to the hypothesis nodes passed as evidence.
2) With all nodes passed as evidence.

The training set consisted of 10000 “and” images and the validation set consisted of 2000 “and” images. The results obtained are described in section 3.

2.3) Task 3: Deep Learning Inference

The aim of this task is to use Deep Learning models such as Siamese Networks and AutoEncoders to predict whether two “and” images belong to the same writer.

Siamese Deep Learning Network

A Siamese Deep Learning Network is a neural network architcture with two or more identical subnetworks. Each subnetwork has the same parameters and weights. Each update on a parameter is updated across each subnetwork. Using these neural networks we can pass each parameter. The output of the two neural networks is passed to a third network which will process this inputs and produce a final output. In our case, we pass two instances of “and” as inputs to the two networks and then obtain the output on whether they are similar or not.

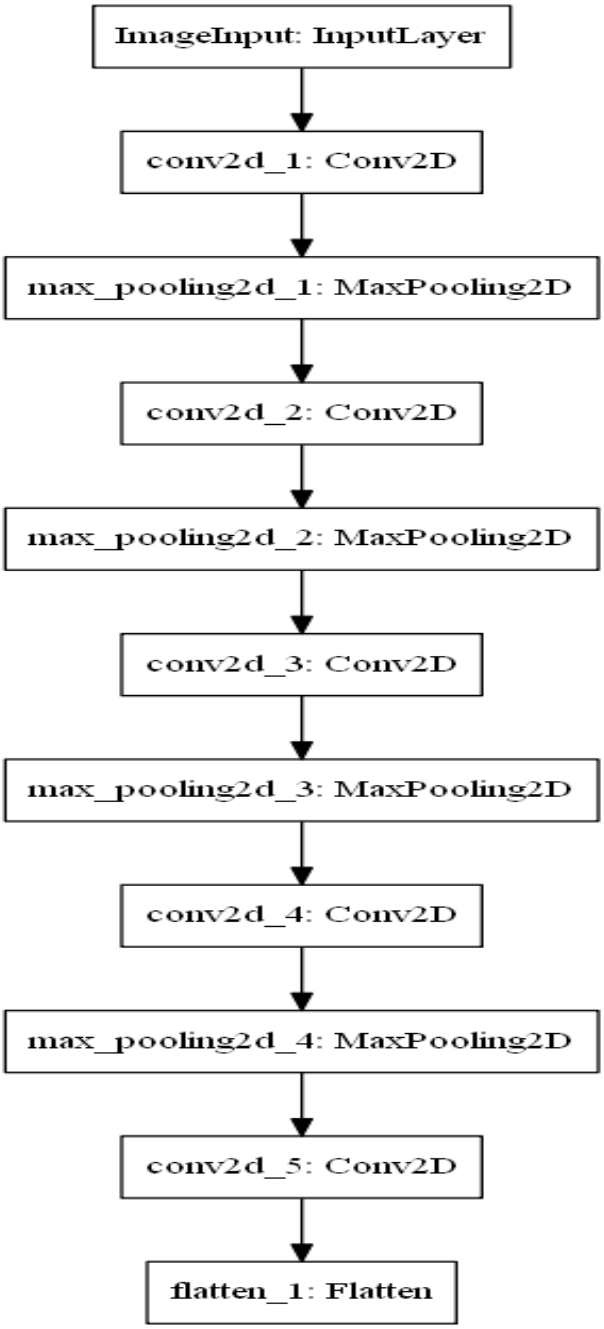

Figure 5: Convolutional Neural Network to process Input Images.

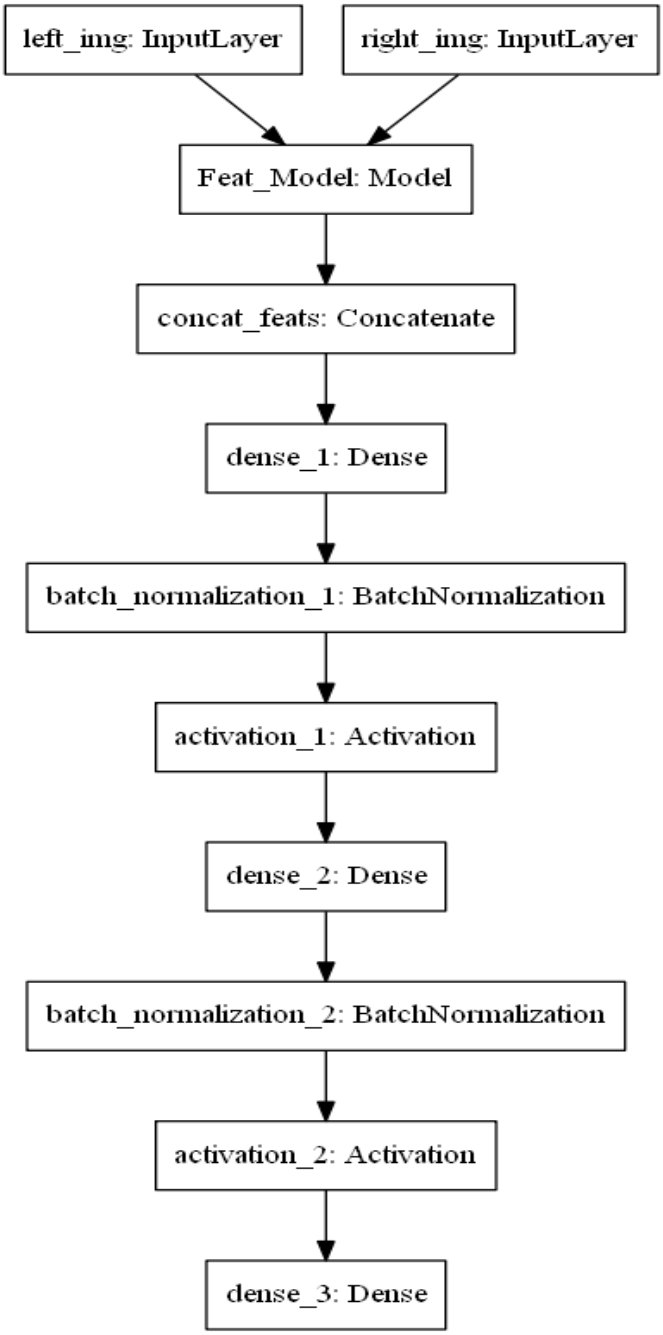

Figure 6: FCN to concatenate and process two images.

The two images are resized to 64x64 and passed through two Convolutional Neural Networks(CNNs). The CNNs are used to reduce the dimensionality of the two input images for the neural network. The two images are concatenated in a layer and then passed through a Fully Connected Network(FCN) which is used to perform transformation on the concatenated values. We evaluate the accuracy and validation loss for both the Training and Validation datasets for Seen, Unseen and Shuffled writers. The loss function used is Categorical Cross

Entropy with Adam optimizer. We run the Siamese network for 10 epochs for each of the three different writer datasets. The number of steps for each epoch is determined by the size of the dataset divided by the batch size used in the Datagen.

AutoEncoder

AutoEncoder are feed forward Neural Networks which produce output same as the input. They perform dimensional reduction on the input and perform reconstruction of the input at the output. The autoencoder consists of three parts: The encoder, the code and the decoder.

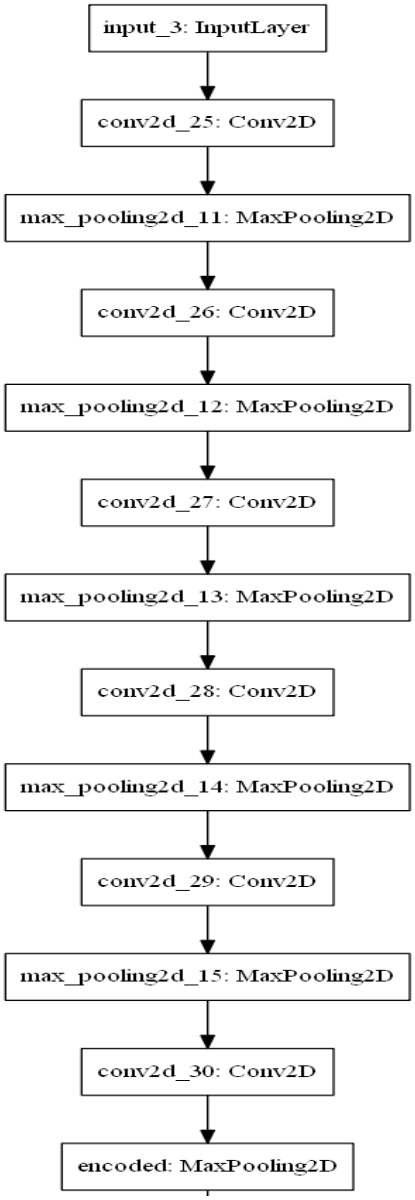

Figure 7: Encoder

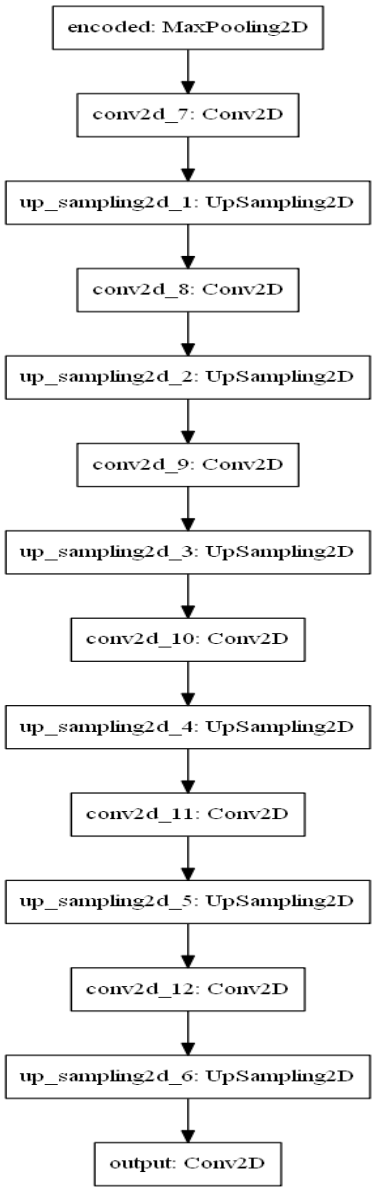

Figure 8: Decoder

We train the autoencoder on Seen, Unseen and Shuffled writer training datasets and use the validation dataset for evaluation. Since, the target image provided by the output is same as the input, we pass the training data to the autoencoder. In order to evaluate our AutoEncoder we simply pass the validation set data as the input to the autoencoder and evaluate the performance of the AutoEncoder. If the autoencoder gives an output similar to the provided input, it is working correctly. We use Binary Cross Entropy as the loss function. We experimented with Stochastic Gradient Descent with weight decay & Adam optimizers of which we found out that Adam optimizer gave a better output. The output of the AutoEncoder is lossy.

In order to obtain similarity between the input and output images by using the values obtained from the latent layer of the AutoEncoder. The values of the latent representations are compared with each other using cosine similarity. Using cosine similarity, we obtain Precision, Recall, F1 Score and Accuracy (Inter and Intra Writer).

Datagen

Passing the entire dataset to the DL models mentioned above may result in the model overfitting to the data. Also, using the entire dataset is also very memory consuming. Hence, it is better to the pass the dataset to our encoder in batches. In case of both the networks we use a batch size of 64. The datagen continually passes predetermined batches of the dataset to the networks for training the networks.

2.4) Task 4: Explainable AI

As Machine Learning models become more complex, the task of producing an interpretable version of the model becomes more difficult. Hence, it is necessary to generate a model which gives an accurate output, but also explains why we got the output.

In order to do so, we retain the weights obtained by the AutoEncoder. We use the same autoencoder model used in Task 3. We remove the Decoder part of the AutoEncoder used in Task 3. The layers of the Encoder are made untrainable in order to prevent any new inputs from altering the weights. The code obtained from the Encoder is passed to 15 different neural networks each of which is designed to give output for the feature it is related to. Each individual neural network will give an output based on the 15 values mentioned in Section 1. We compare the values provided to us in the Seen, Unseen and Shuffled Writer datasets with the ones obtained from the Frozen Encoder. Finally, with the real and predicted outputs which we have, we compute the cosine similarity for each real and predicted value we have obtained. The cosine similarity is then used to obtain the similarity scores for the 15 different features available in the dataset.

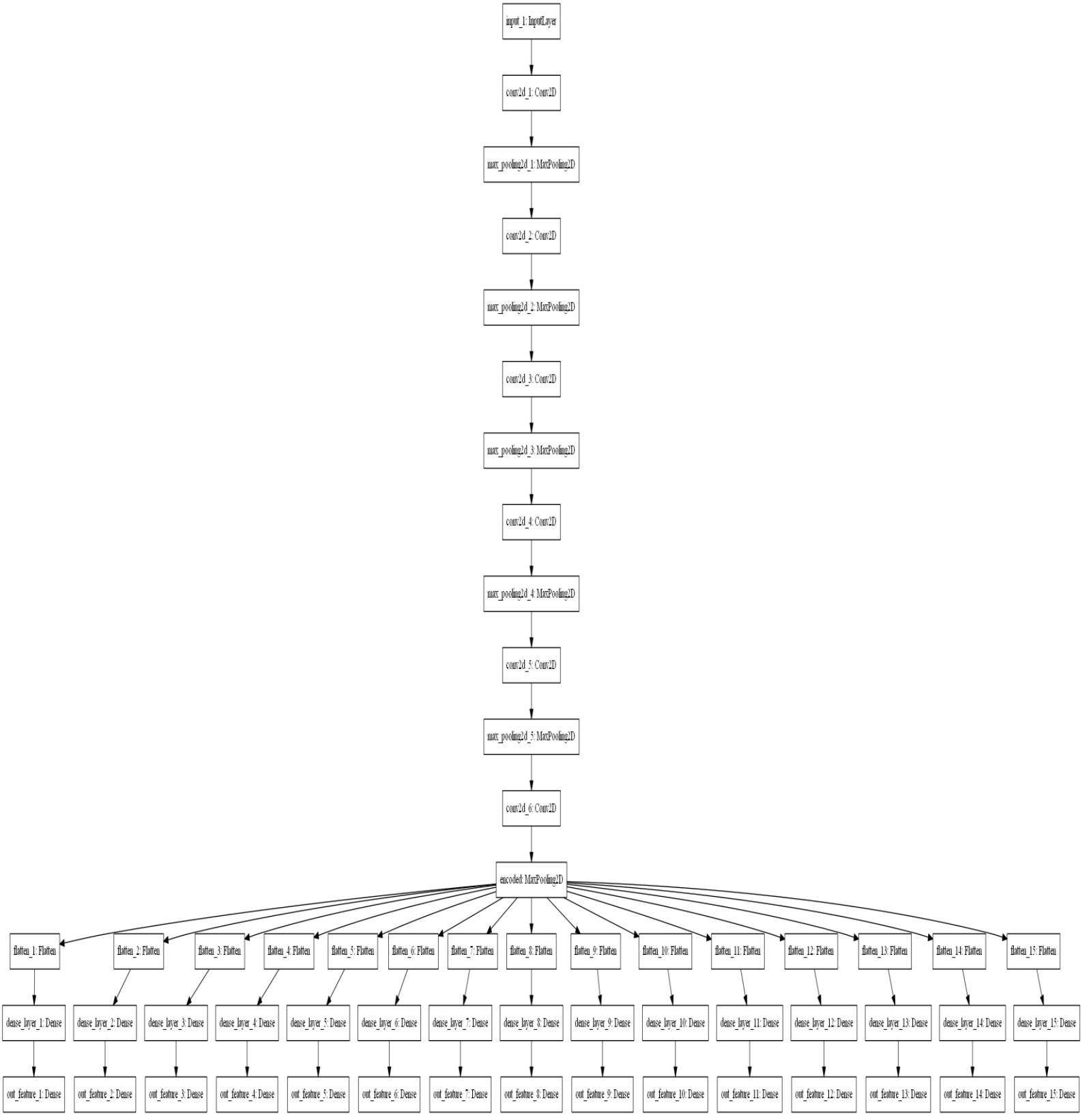

Figure 9: Frozen Encoder

3. Results

Bayesian Network

Number of edges of HillClimb Search Model: 42
Number of edges of Concatenated Model: 88

a) Bayesian Network with all nodes passed as evidence

Training Accuracy	Time Taken (in hours)	Testing Accuracy	Time Taken (in hours)
78.61%	1.030	78.1499999999%	0.209

b) Bayesian Network with only nodes directly attached to hypothesis nodes as evidence

Training Accuracy	Time Taken (in hours)	Testing Accuracy	Time Taken (in hours)
65.44%	2.852	65.75%	0.58

Siamese Deep Learning Network

a) Siamese Deep Learning Network (Seen Dataset)

Number of Epochs	Training Accuracy	Validation Accuracy
2	0.9977	0.5009
10	0.9998	0.5002

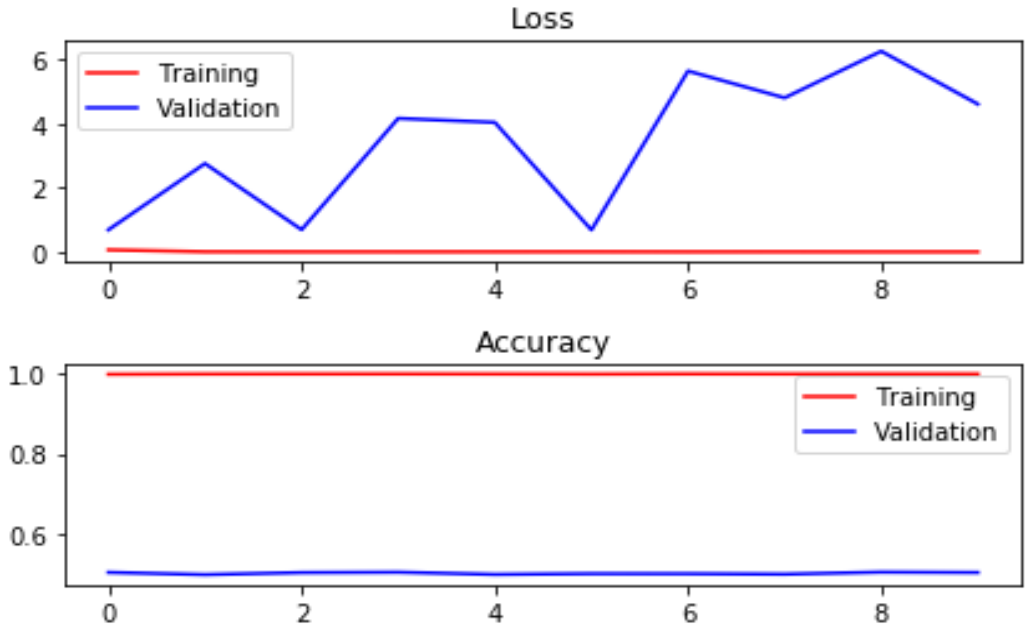

Figure 10: Accuracy and Loss for Training and Validation Seen Datasets

b) Siamese Deep Learning Network (Unseen Dataset)

Number of Epochs	Training Accuracy	Validation Accuracy
2	0.9635	0.4971
10	0.9997	0.5387

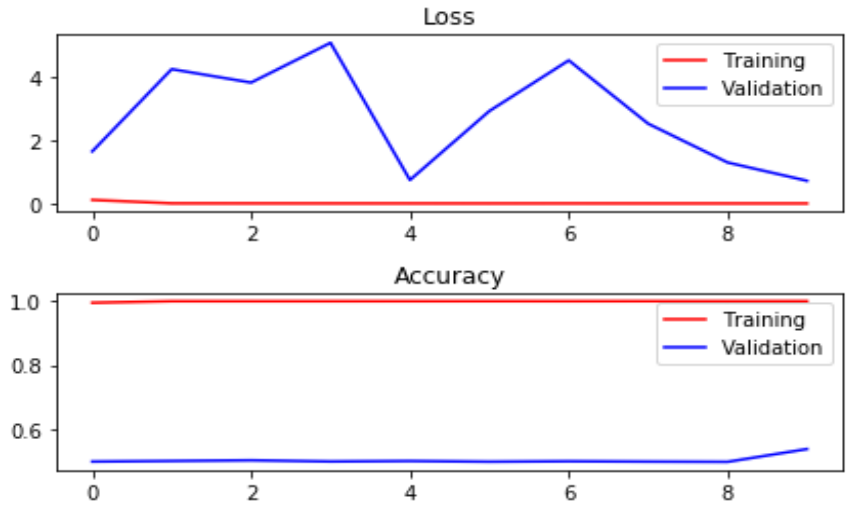

Figure 11: Accuracy and Loss for Training and Validation Unseen Datasets

c) Siamese Deep Learning Network (Shuffled Dataset)

Number of Epochs	Training Accuracy	Validation Accuracy
2	0.9996	0.5009
10	0.9820	0.5021

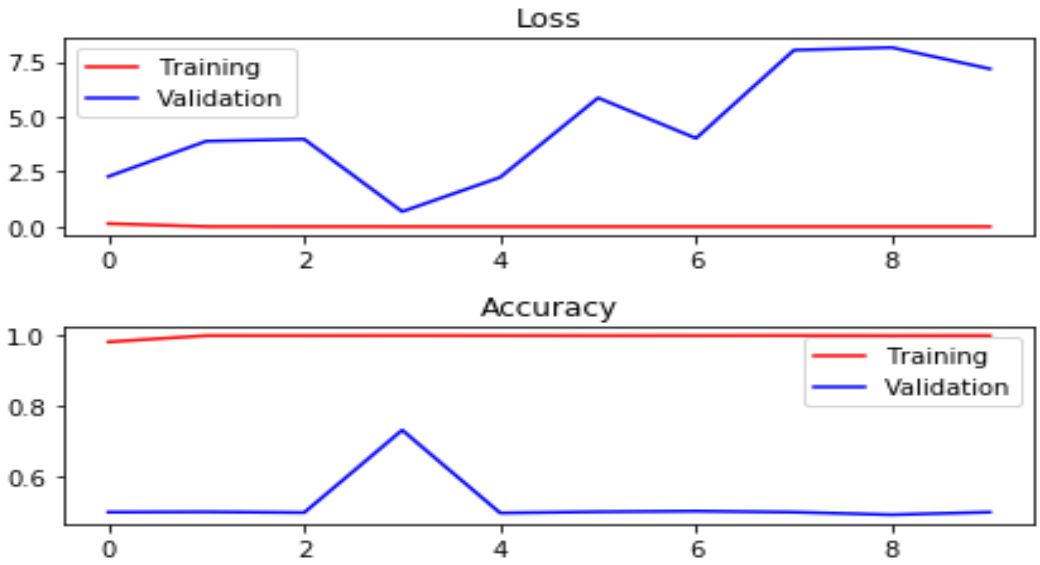

Figure 12: Accuracy and Loss for Training and Validation Unseen Datasets

AutoEncoder

As demonstrated by the results above, we see that the Siamese Deep Learning Network tends to over-fit to the training data. As a result of this we obtain a poor validation accuracy. Hence, we change our model and use AutoEncoder and eventually Frozen AutoEncoder which provides better results as described below.

a) AutoEncoder (Seen Writer Dataset)

Cosine Threshold	Precision	Recall	F1 Score	Intra Writer Accuracy	Inter Writer Accuracy	Accuracy
0.66	0.025027	0.83365	0.048595	0.6033842	0.92503	0.926902

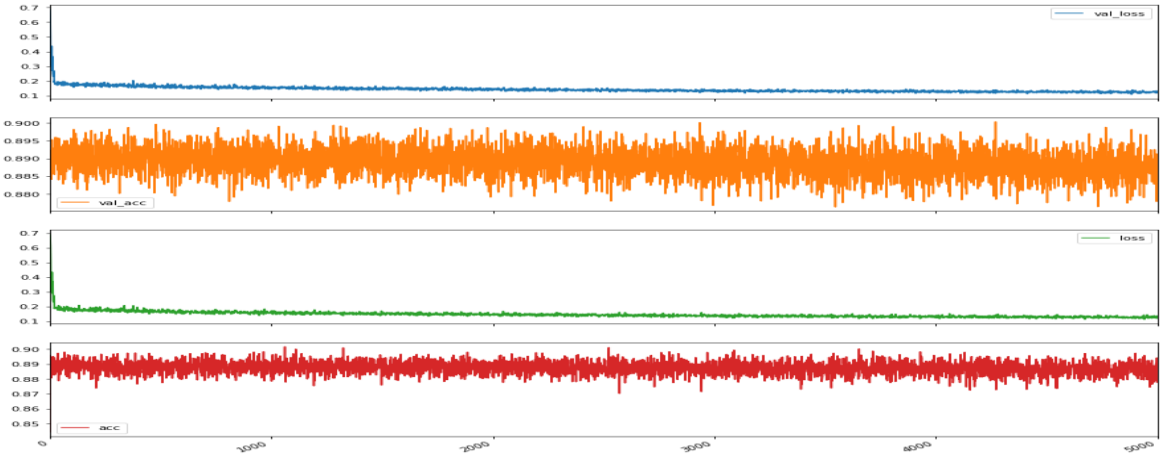

Figure 13: Loss and Accuracy for AutoEncoder with Seen Writer Training and Validation Dataset

b) AutoEncoder (Unseen Writer Dataset)

Cosine Threshold	Precision	Recall	F1 Score	Intra Writer Accuracy	Inter Writer Accuracy	Accuracy
0.66	0.097869	0.5904275	0.167906	0.6033842	0.928429	0.9349889

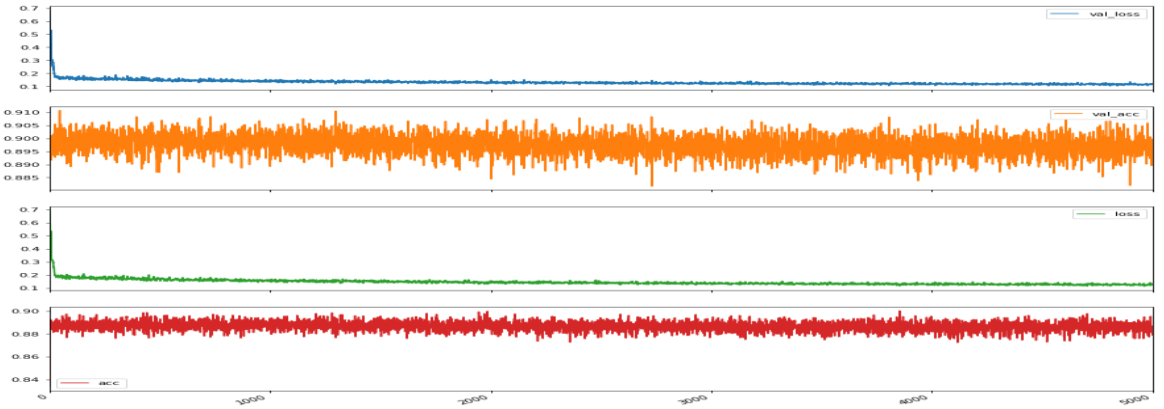

Figure 14: Loss and Accuracy for AutoEncoder with Unseen Writer Training and Validation Datasets

c) AutoEncoder (Shuffled Writer Dataset)

Cosine Threshold	Precision	Recall	F1 Score	Intra Writer Accuracy	Inter Writer Accuracy	Accuracy
0.66	0.0194463	0.74569	0.0379042	0.7798338	0.92123	0.922753

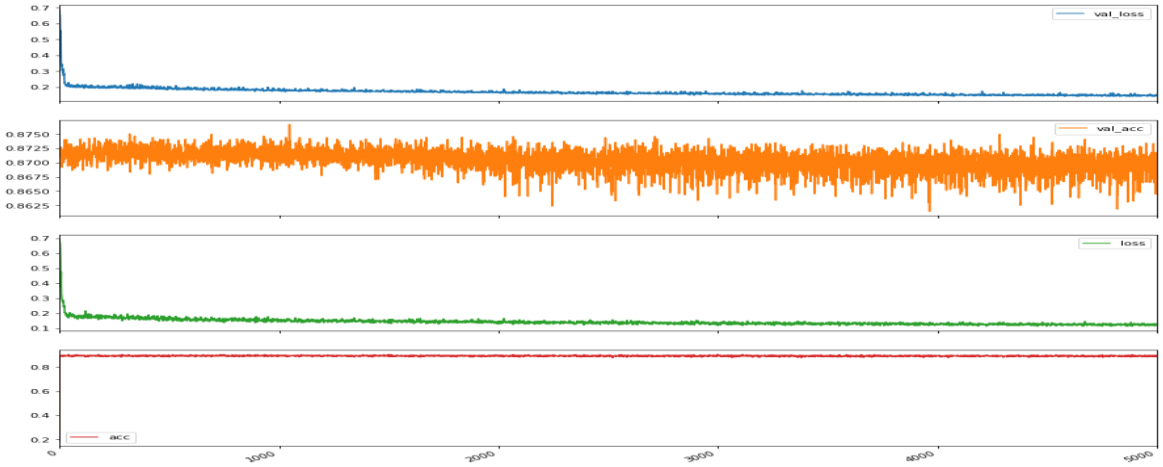

Figure 15: Loss and Accuracy for AutoEncoder with Seen Writer Training and Validation Datasets

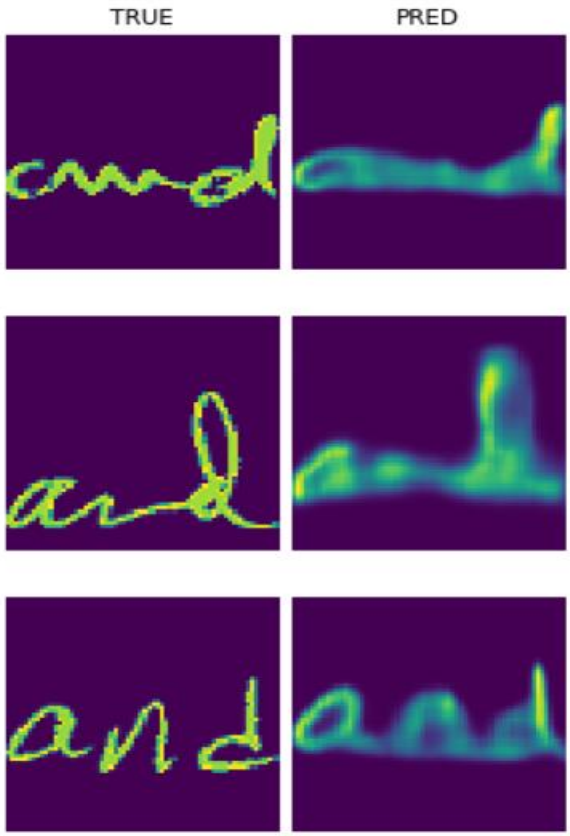

Figure 16: AutoEncoder Input and Output

Frozen AutoEncoder

Frozen AutoEncoder (Seen Writer Dataset)

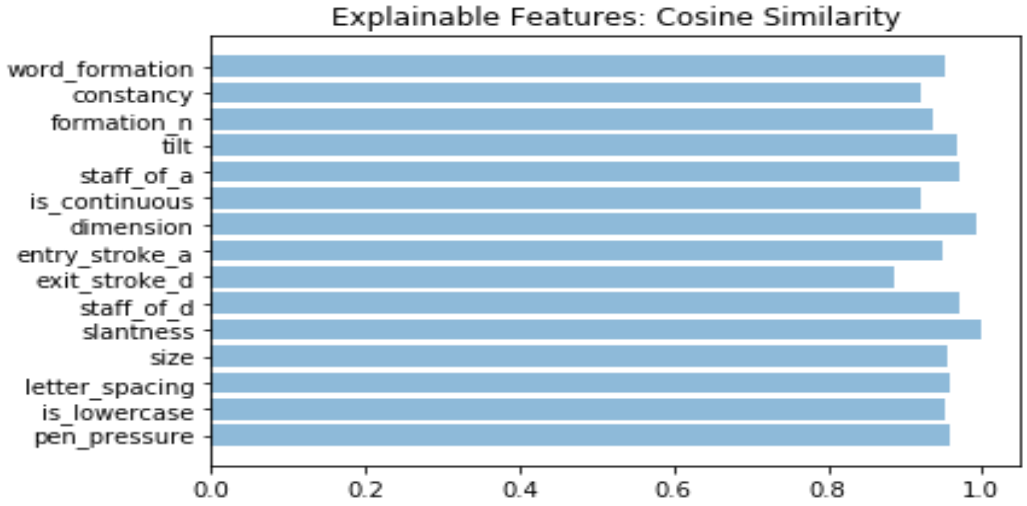

Figure 17: Similarity Score Seen Dataset

Frozen AutoEncoder (Unseen Writer Dataset)

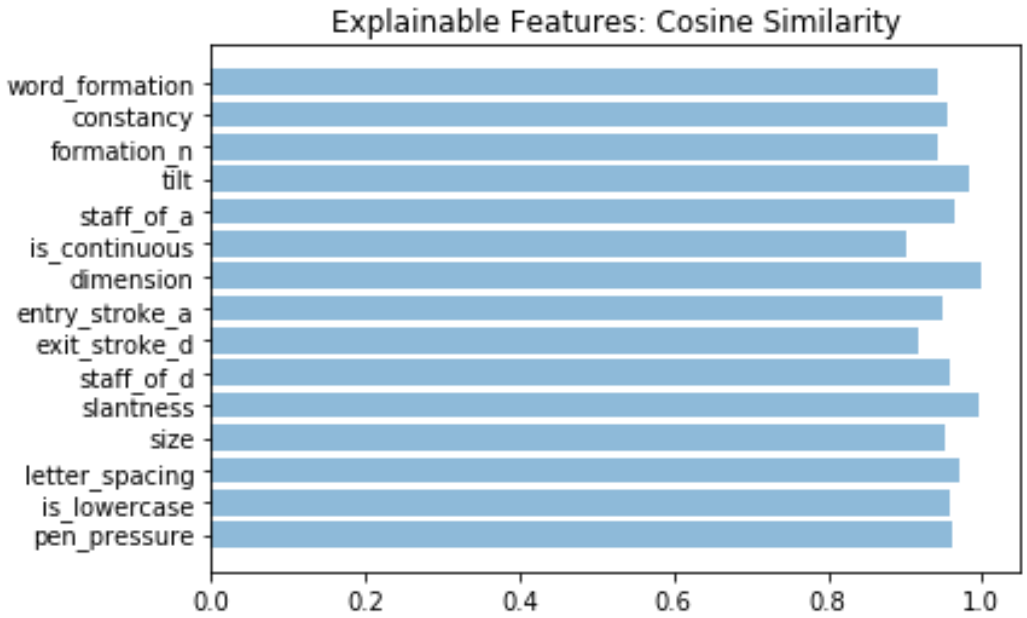

Figure 18: Similarity Score Seen Dataset

Frozen AutoEncoder (Shuffled Writer Dataset)

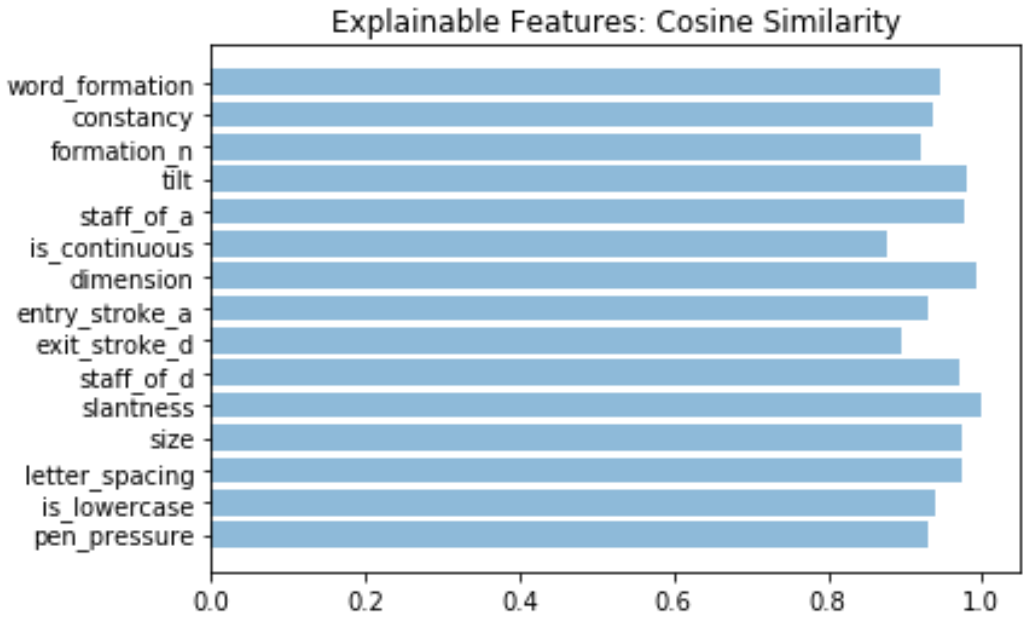

Figure 19: Similarity Score Shuffled Dataset

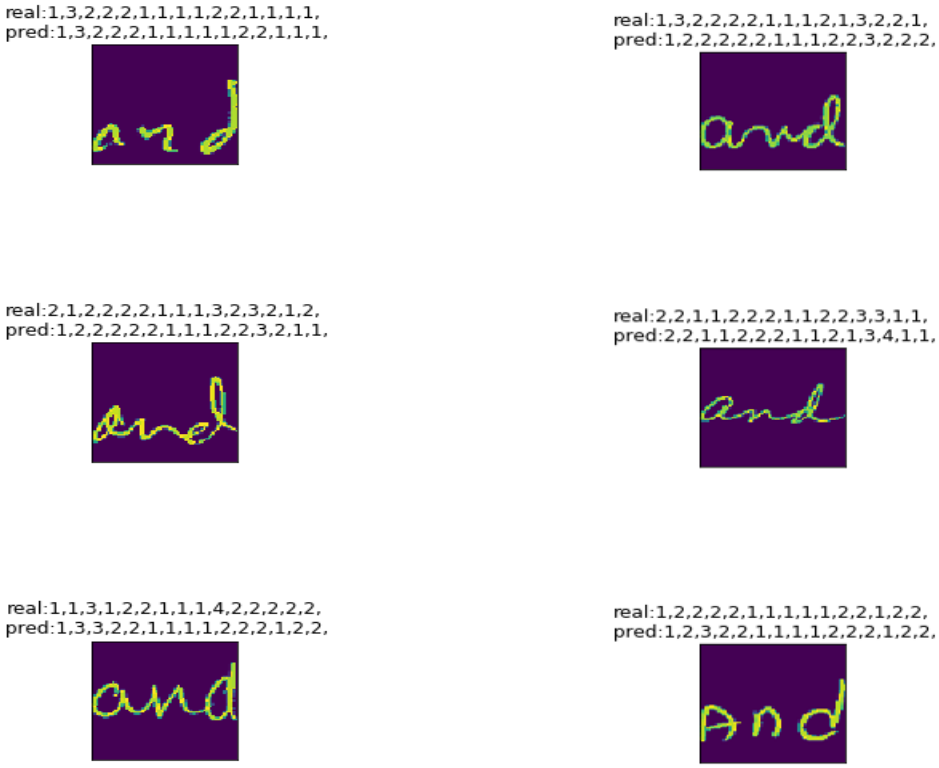

Figure 20: Features provided as output from the Frozen Encode

Conclusion:

As models become more complex, the task of producing an interpretable version of the model becomes more difficult. The lack of transparency in the output of these models is a problem for the people who want to understand the way the model works to help them improve their service. Hence, it is necessary to develop models which are not only accurate but also sufficiently conclude why they are accurate.

References:

- http://sujitpal.blogspot.com/2017/02/using-keras-imagedatagenerator-with.html
- https://medium.com/@ensembledme/writing-custom-keras-generators-fe815d992c5a
- https://medium.com/@neerajsharma_28983/intuitive-guide-to-probability-graphical-models-be81150da7a
- https://medium.freecodecamp.org/an-introduction-to-explainable-ai-and-why-we-need-it-a326417dd000
- https://github.com/mshaikh2/HDL_Forensics/blob/master/AML_SPRING_2019/MultiTask/Multitask-FrozenEncoder-seen.ipynb
- https://github.com/mshaikh2/HDL_Forensics/blob/master/AML_SPRING_2019/AutoEncoder/DL_AE_Maxpool_1x1x512_Seen.ipynb