Explainable AI - Writer Verification

Pratik Pravin Kubal
Department of Computer Science
University at Buffalo
pkubal@buffalo.edu

Abstract

In this paper, we explore the creation of explainable AI for the task of Writer verification. We use techniques such as Probabilistic Graphical Models, Autoencoders, Siamese Networks, and Multi-Task Learning to create an interface for explainable interace for 15 handcrafted features. We also explore a distant learning based method and Sigmoid Structured CPD tables or networks.

1 Introduction

Writer Verification consists of given two samples infer or decide if they are from the same writer or not. Various architectures already exist for this task, Refer 2. However, explaining why particular features were selected is difficult, this is the problem of Explainable AI. In this work, we explore some techniques such as PGM, Siamese Networks, and Autoencoders combined to create an explainable interface for inferring the categories. We also explore some of the problems while creating an explainable AI system.
We also explore Sigmoid Structured CPDs and Entropy-based Distant Supervision for Probabilistic Graphical models. We hope to extend the work in 2.

2 Relevant Work

There has been extensive work in the field of writer verification by Mohammad Abuzar Shaikh, Mihir Chauhan, Jun Chu, and Sargur Srihari in the paper Hybrid Feature Learning for Handwriting Verification. In the paper, they combine the features generated by deep learning architectures such as CNN, Autoencoders and Siamese networks with Human engineered features such as SIFT. Although these approaches could be used for writer verification they lack explainability.

3 Methods

Limited by the size of the document some of the graphs such as Tensorboard graphs for multitask network couldn't be accommodated. Refer Github[1].

3.1 Annotations

Annotations were divided amongst a class of approximately 160 students, each receiving around 150 images for annotating. There were 15 categories for each image and they were graded heuristically. Separately, over the spring break the course TA extended the annotations to approximately 12000 images. The motivation for Annotators to do well in the task was primarily to score high in the project, we do not know the quality of annotations, however, we assume that the quality is at least above average.

[1] Github Code: https://github.com/pratik-kubal/AML-XAI-2

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

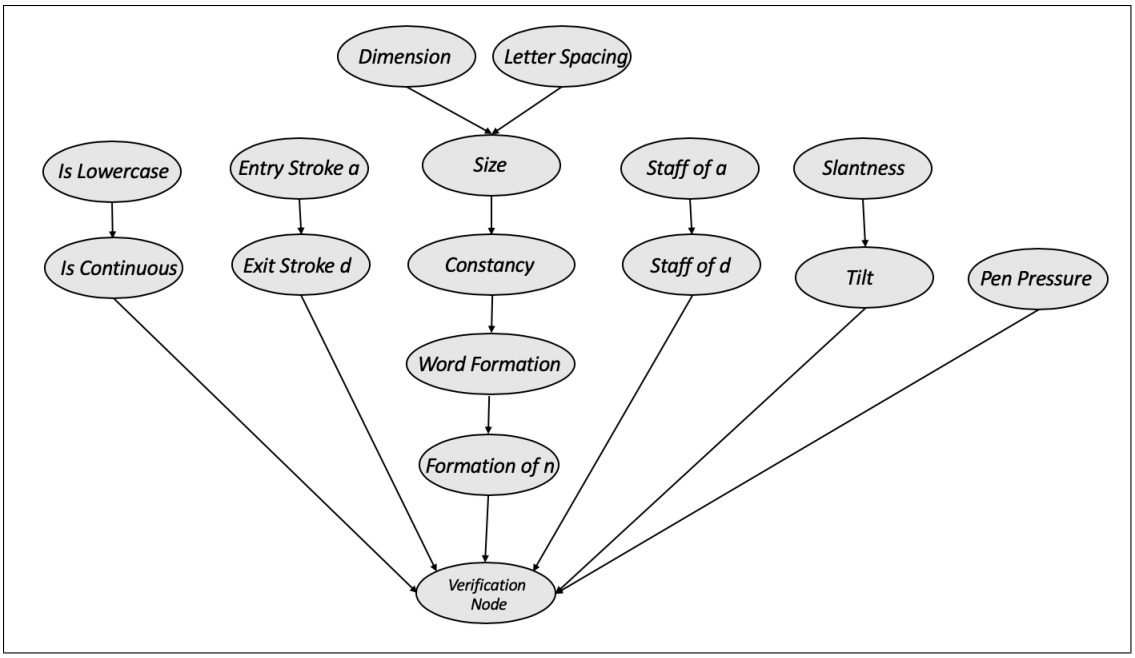

Figure 1: Bayesian Network for Writer Verification

3.2 Writer Verification by Probabilistic Graphical Models

Initially, experimenting with Hill Climbing and Naive Structure Learning approaches weren't advantageous – they were plagued by taking too long for CPD estimation (Structure Learning) and high time for inferences. Moreover, abandoning structure and connecting all the nodes to the verification node gave good results but the time to inference was too high.
Mihir suggested an intuitive structure, given in Figure 1. Each node contains two observed variables of the two writers connected towards an unobserved variable of similarity. The CPDs are hand crafted based on basic intuition of the probability distribution. Each unobserved similarity nodes are connected towards next in a chain and the verification node. This structure makes it possible to infer on all the variables for both writers. The internal structure can be seen in 2.
For the verification node, we use a Sigmoid CPD, which is discussed in 3.2.1. This method was discussed by Daphne Koller.
While training, we have also found out distantly supervised models with Sigmoid CPD work well. This method is discussed in 3.2.2.

3.2.1 Deterministic Sigmoid CPD

The deterministic CPD given in Figure 3 is an example where each feature is an observed variable connected to unobserved nodes Z_i. Each connection has a node potential W. The Bias or intercepts node is connected to a node which sums the input and applies sigmoid function deterministically. The result is the probability of the inference.
We use Logistic Regression to learn the weights and Bias over for the data. The input nodes are similarity nodes. We train the logistic regression model on the inference of these nodes. After finding the weights and intercept we create functional deterministic nodes.

3.2.2 Entropy Based Distant Supervision

Even after considering similarity nodes, the time to run inference is still significant. As distant supervision, we can pick up top N positive and negative label pairs having high entropy. Inference can be run on these small data set and then we can construct the Deterministic Sigmoid CPD structure. In our experimentation we have found out it gives a reasonable F1 Score. More discussed it in Section 4.

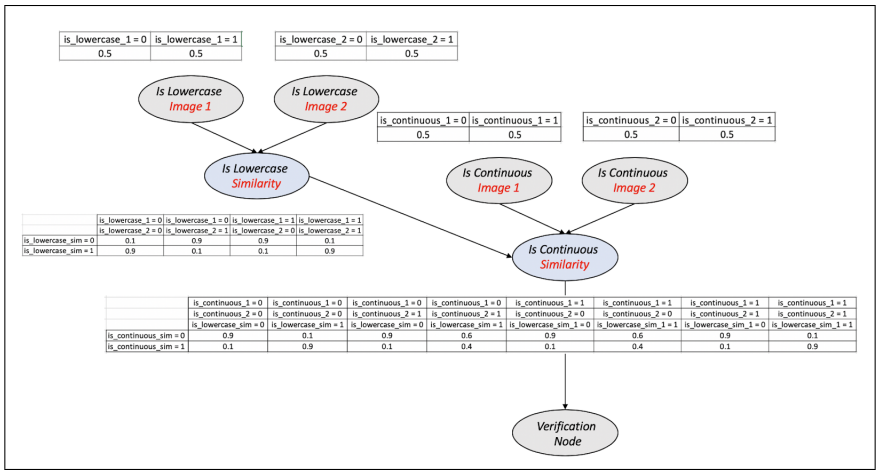

Figure 2: Network intuition

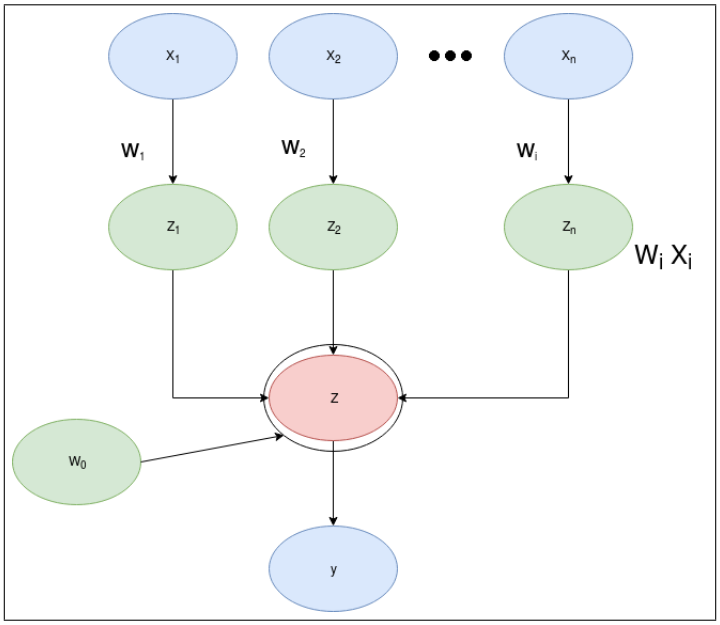

Figure 3: Sigmoid Network

3.3 Writer Verification by Deep Learning

Siamese networks are two identical layer networks with shared weights between them. Both networks are trained simultaneously and each learns to minimize the loss according to the difference in images. While being powerful in areas of transfer learning, in our limited time, we couldn't find perfect hyperparameters which make it work. Discussed further in 4.
Mohammad's approach for Autoencoders works better, which we think is due to the auto-encoder decomposing the image into latent features. After an auto-encoder is trained on images, the decoder can be separated to work with the latent features. Distance measures such as cosine similarity or Euclidean Distance can be applied for the two writer images.
Both approaches Siamese and Encoder based writer verification employ distance measures, however more often than not these distance measures are thresholded. Instead of arbitrarily choosing a threshold or keeping it as 0.5 we try to balance the precision and recall on the latent features distance.

3.3.1 Siamese Loss Function

Contrastive Loss is usually used for Siamese network. At each update, Contrastive Loss tries to bring two input are similar tensors then it will try to bring them as close as possible. Another approach which we can use is creating a dense layer with sigmoid and then using binary cross-entropy as loss function with an obvious choice of binary accuracy as the metric. However, this isn't the case with a contrastive loss, contrastive loss requires a distance measure such as cosine distance or Euclidean Distance. The Cosine distance is simply the dot product of two vectors, each of the distance measures have their own advantage. Such as cosine distance or cosine similarity is better in a higher dimensional space it gives better variability between two vectors.

3.4 Explainable AI

Usual algorithms lie on a high dimensional space in terms of explainability and learning performance. Usually, high performing models are black boxes which do not give explanation of their decisions. Explainable AI as the name suggests explains its actions. Its an ongoing area of research about whether the making models explain their decisions decrease the performance, we hope to test this debate in this work.
For Explainable AI we are going to use three modules. firstly we are going to learn the latent representations of the images. These features in latent space will then be mapped to a Probabilistic Graphical Model using a multitask network.

4 Experimentation

Seen data set gives the network to learn writing samples of all the writers, we use it to create a model and test various hyper-parameters on it. Therefore, given below unless specified most of the experimentation covers seen dataset.

4.1 Bayesian Network

Initially, we started with Naive Hill Climbing based approach for Structure learning. However, if we observe a node, the flow of inference stops, so in a connected graph with only observed variables at least one node would be not used. Moreover, this best case scenario requires a high number of indegree which takes a toll on time to train and time to infer. Some preliminary results with Time-connection trade-off are given in 1.
The second method which we experiment on is first finding structure using indegree 1 and then augment it with another set of features. In the end, we use MLE to find the CPDs associated with the nodes. The accuracy is low because the network is sparsely connected.
The third approach is a naive feature subtraction method, it is based on a similar project done last semester for the same data set although discriminative method showed good results.
The last approach is by similarity nodes connected to the verification node. The main bottlehead in this is still running inferences, possible alternatives could be running approximate inference algorithms. However, here we use Distantly Supervised Pairs and find reasonable results. The results are summarized in Table 1. The main advantages of distant supervision are that the train time isn't dependent on the data, rather the quality of data, however, the inference time for testing is still dependent on the scale of data.
One problem which we have found in the distant supervision approach is that the values easily overfit, maybe future research can cover these aspects. We tried using l2 regularizer but it didn't affect the over fit that much.
Due to training time constraints and high resources required for AWS, we use the entropy-based approach for Shuffled and Unseen data sets.
Using the approach we were able to reach 76% binary accuracy for the Seen data set.

4.2 Writer Verification by Deep Learning

We tried creating five different networks for Siamese Network but were not able to go beyond 50% without over-fitting. First, it seemed that the network was either dense or sparse, but it seems that while train data being complete the validation data seems incomplete. Around 900 rows seem to have less variability. After not getting adequate results in Siamese network we used Autoencoder

Table 1: Writer Verification by PGMs for Seen Dataset

Method	Max Indegree	No. of Conns	Time to Train	Time to Infer	Accuracy
Basic	2	43	~ 1 Hour	~45 mins	~76
Augmentation	1	22	~ 5 min	~45 mins	~ 56
Feature Subs	2	28	~ 3min 39s	~ 5min 28s	~ 77
Sigmoid CPD	3	50	~ 3hours	~2mins 30s	~ 73
Sigmoid CPD wt Entropy	3	50	~ 2mins	~ 2mins	~ 66

Table 2: Writer Verification accuracies by entropy sigmoid CPD

Dataset	Precision	Recall	Train f1	Validation f1 difference
Seen	0.71	0.821	0.76	0.098
Shuffled	0.73	0.79	0.76	0.16
Unseen	0.66	0.98	0.79	0.15

from the explainable AI module. The autoencoder is already trained, see Section 4.3. Initially, we tried freezing the layers and trying out Cosine Distance based Lambda Layer with Contrastive loss, the same with the trainable encoder. The results were not that good, the F1 score was low. Then we turned off all the weights and tested out a thresholding based lambda layer on the threshold found in 4.3, however, still the model didn't improve. To verify this we manually found cosine distance by sklearn package and the threshold which is used to balance it, surprisingly the threshold was 94 which contrasts with the results found in the XAI module. Moreover, the lambda layer based on a threshold is not differentiable – unlike sigmoid – so there is no use in training the model since the gradients won't backpropagate. The best approach we found by experimentation is finding the cosine distance by sklearn and deciding the threshold based on the data such as seen, unseen and shuffled.
While working with the validation file we couldn't get any good results. All the results in spite of changing models or hyperparameters either overtrained. While controlling overtraining the maximum accuracy we found was 52% and minimum training loss was 0.6912.
We created a secondary validation data based on all the permutations of the writers are using a 0.98 cosine similarity threshold we were able to get 0.68 as the f1 score. The results of such thresholding are given in 3. We try to balance the precision and recall, we find that 0.98 balances it.

4.3 Explainable AI

After training, the autoencoder converged towards 0.006 loss, the predictions from the autoencoder can be seen in Figure 6. Refer table 4 for performance on other datasets.
The multitask network is made up of the trained encoder. We add a Dense 128 layer to the latent

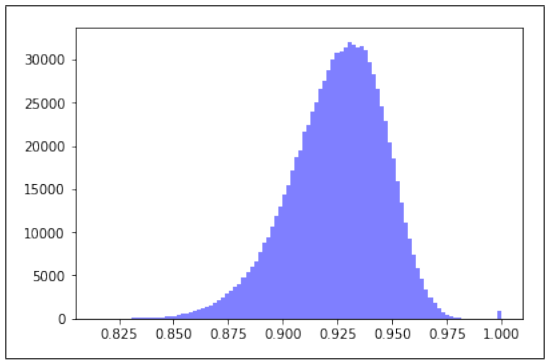

Figure 4: Writer Cosine Similarity Distribution (On x-axis the cosine similarity, on y-axis the count)

Table 3: Writer Verification Cosine threshold by Deep Learning for Seen Dataset

Cosine Threshold	Precision	Recall	f1
0.97	0.06	0.7	0.11
0.98	0.62	0.56	0.59
0.99	0.99	0.51	0.43
0.985	0.95	0.53	0.68

Table 4: AutoEncoder

Dataset	Training Loss	Validation Loss
Seen	0.062	0.058
Shuffled	0.068	0.071
Unseen	0.0653	0.059

feature layer in the encoder section of the autoencoder. Then we add a dense layer with a number of categories as softmax activation function mapping for each feature and then compile the model with categorical_crossentropy and categorical_accuracy. We found out that rmsProp algorithm with Learning Rate 0.001 works best amongst different settings. For the seen dataset we evaluated for around 30 different runs with different hyperparameters, the tensorboard logs can be found on github.
After training the multitask model on training data we attach the PGM created in Section 3.2 as an output of the multitask prediction. We create a function to handle this task.

5 Observations

We see that the time for inference has not changed by a significant margin, however, the accuracy has surely increased. Moreover, compared to the previous approach we can now use all the variables in our inference query since each observed variable is connected to at least 1 unobserved variable. Entropy approach is also a quick approach, but the data which needs to be selected should be carefully selected or else there could be an over-fitting of data. Moreover, we can see that the CPD based approach perfectly finds the weights of the features, it is a mix of generative and discriminative approaches. While primarily being generative, it estimates the node potentials using a discriminative approach.
The Task of writer verification through Deep learning networks was difficult, no matter what approaches we used either the model was overtraining or running at 50% accuracy. The problem might be in the data because when used cosine similarity we get a good f1 at around 68. This data was generated by pairing each writer with another writer in all possible permutations. Threshold balancing also worked in that case, due to the scarcity of data in the validation file the model fails to train.
Furthermore, in Autoencoders we can see that the low loss models perfectly regenerate the output as seen in 6. Moreover, balancing to balance the recall-precision one might use the area under the distribution of 4 to find a point which balances the curve. Training the autoencoder was relatively simpler than other models.
Referring to Table 1 we can see that in the case of Seen dataset initially the highest writer verification accuracy we can reach is 76%. However, after Explaining module is fitted, the overall accuracy decreases to around 65. This drop might be because of low f1 scores of Multi-Task Learning model. Also, the errors build up over different methods, for example, if autoencoder classifies a latent feature incorrectly, then this leads to the output of multitask and graphical model to suffer. Hence, it makes sense to create short models as possible.

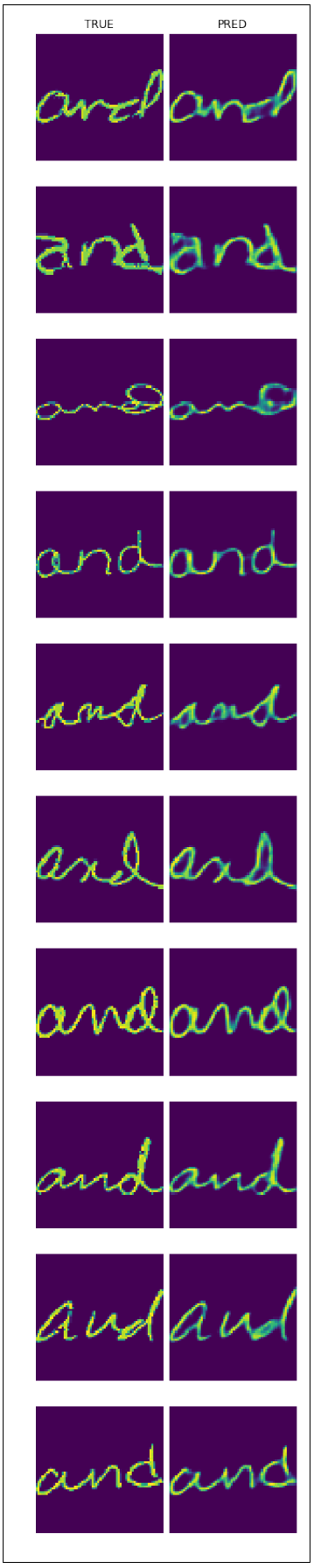

Figure 5: Autoencoder input and output

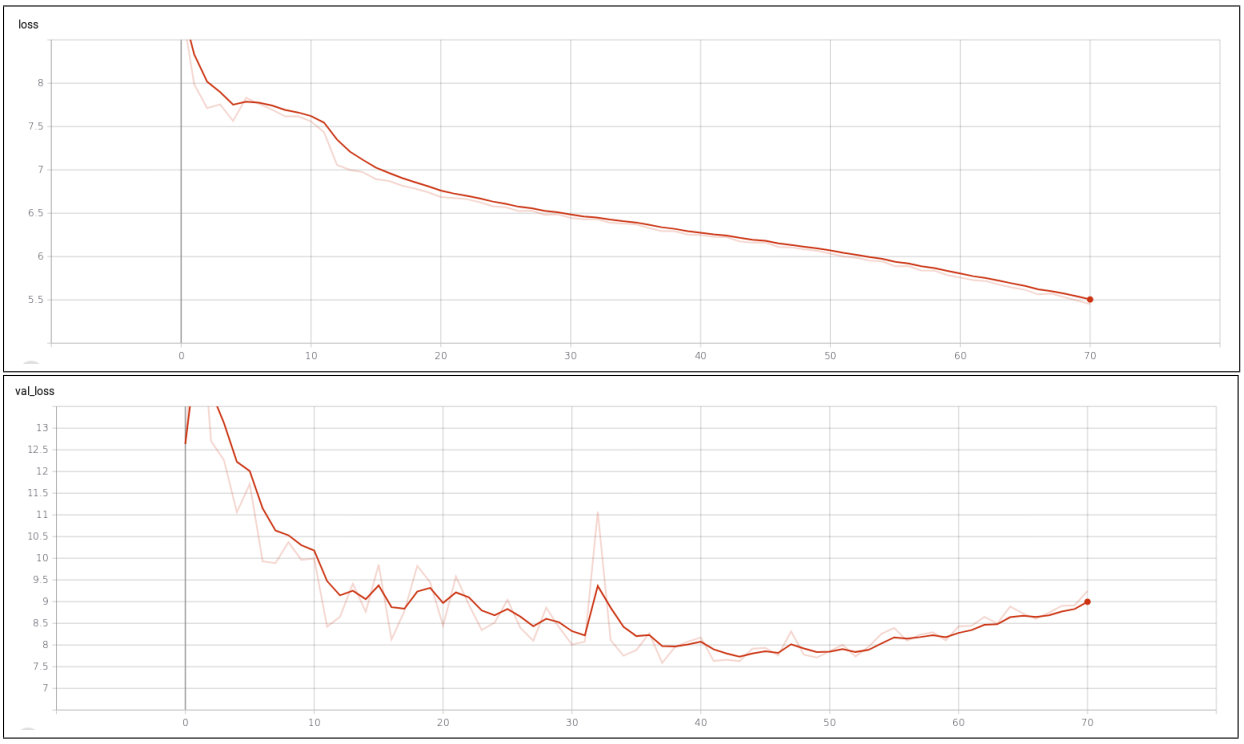

Figure 6: XAI top training Loss and Bottom Validation Loss

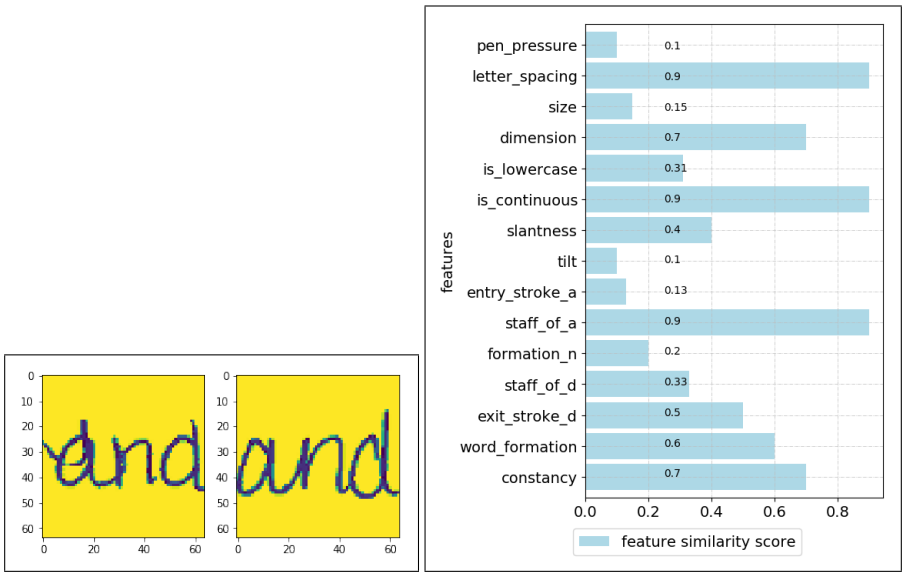

Figure 7: XAI Explaination mechanism

Table 5: MultiTask Network

Dataset	Training Loss	Validation Loss
Seen	5.42	6.99
Shuffled	9.23	9.29
Unseen	8.67	8.826

Table 6: XAI Accuracy

Dataset	Accuracy
Seen	65
Shuffled	53
Unseen	50

6 Results

We can see the interface of XAI in Figure 7.
We find that the accuracy of the model decreases after an explainable interface is attached to it. This is seen in Table 6, we see that all the datasets had a drop in accuracy after multi-task learning is applied. We also find that multitask learning finds it very difficult to find a global minimum. Also, as we have seen from Writer Verification task by Probabilistic Graphical models that handcrafted structure learning gives better results, so future work could put emphasis on the networks which are carefully crafted to provide an explainable interface.
Lastly, we also found out that a Sigmoid CPD gives better results than a Naive Hillclimb search, however talking about training, in this work we were unable to create a Deep learning framework for writer verification. All the models which we experimented on lingered around 50-55% accuracy.

References

[1] Probabilistic Graphical Models Principles and Techniques (2009) Daphne Koller and Nir Friedman, The MIT Press. Retrieved March 08, 2019, from https://mitpress.mit.edu/contributors/daphne-koller