Writer Verification, Explainable AI

Shruti Bendale
Person Number: 50289048,
School of Engineering and Applied Sciences,
University at Buffalo, Buffalo, NY, 14214
shrutitu@buffalo.edu

Abstract

The purpose of this project is to compare handwritten samples of the letter 'and' of two writers and verify if the handwriting samples are of the same writer. Initially, 15 handcrafted features were identified, and the images were annotated for these features. These 15 features of all the images were used to generate a Bayesian model and calculate the conditional probability tables. A combinational Bayesian model with 30 features and one hypothesis node was created to compare two images and generate the output (similar/not similar) at the hypothesis node. We also create a Siamese Convolutional Neural Network to take the handcrafted features of two images and generate the prediction. Finally, instead of having to manually annotate the images, we use an Autoencoder combined with CNNs to generate features for the images.

1. Introduction

The problem of handwriting identification is to interpret intelligible handwritten input automatically, which is of great interest in the pattern recognition research community because of its applications to many fields. As one of the fundament problems in designing practical recognition systems, the recognition of handwritten digits is an active research field. Immediate applications of the digit recognition techniques include postal mail sorting, automatic address reading and mail routing, bank check processing, etc. Most importantly, one of the applications is handwriting verification for digital forensics analysis where we have to identify if two given handwriting samples are by the same writers.

In this project we were given handwritten 'and' samples of images. The project was divided into 4 parts:

a. **Annotations:** The first task was to annotate the given images based on the given 15 features.

b. **Probabilistic Approach:** The second task was to use these labelled features to create a combined Bayesian Network for two images and that can be used to identify the similarity of the images.

c. **Deep Learning Approach:** The third task was to get the similarity of two images using a deep neural model. For this purpose, we use the Siamese Convolutional Neural Network approach to predict the labels.

d. **Representation Learning Approach:** The final task uses representation learning to learn the features from the images. We then convert these features to human explainable features to get the features in terms of the 15 characteristics we had identified.

2. Implementation

Task 1: Annotations

An image of 'and' sample was characterized by identifying 15 explainable features:

1. Pen Pressure	2. Letter Spacing	3. Size	4. Dimension
5. is_lowercase	6. is_continuous	7. Tilt	8. Slantness
9. entry stroke of 'a'	10. Staff of 'a'	11. Staff of 'd'	12. Exit stroke of 'd'
13. Formation of 'n'	14. Word formation	15. Constancy	

All the images were annotated for these features by human annotators using the 'CEDAR Handwriting Truthing Tool'. An example of an image and its human observed features is given below:

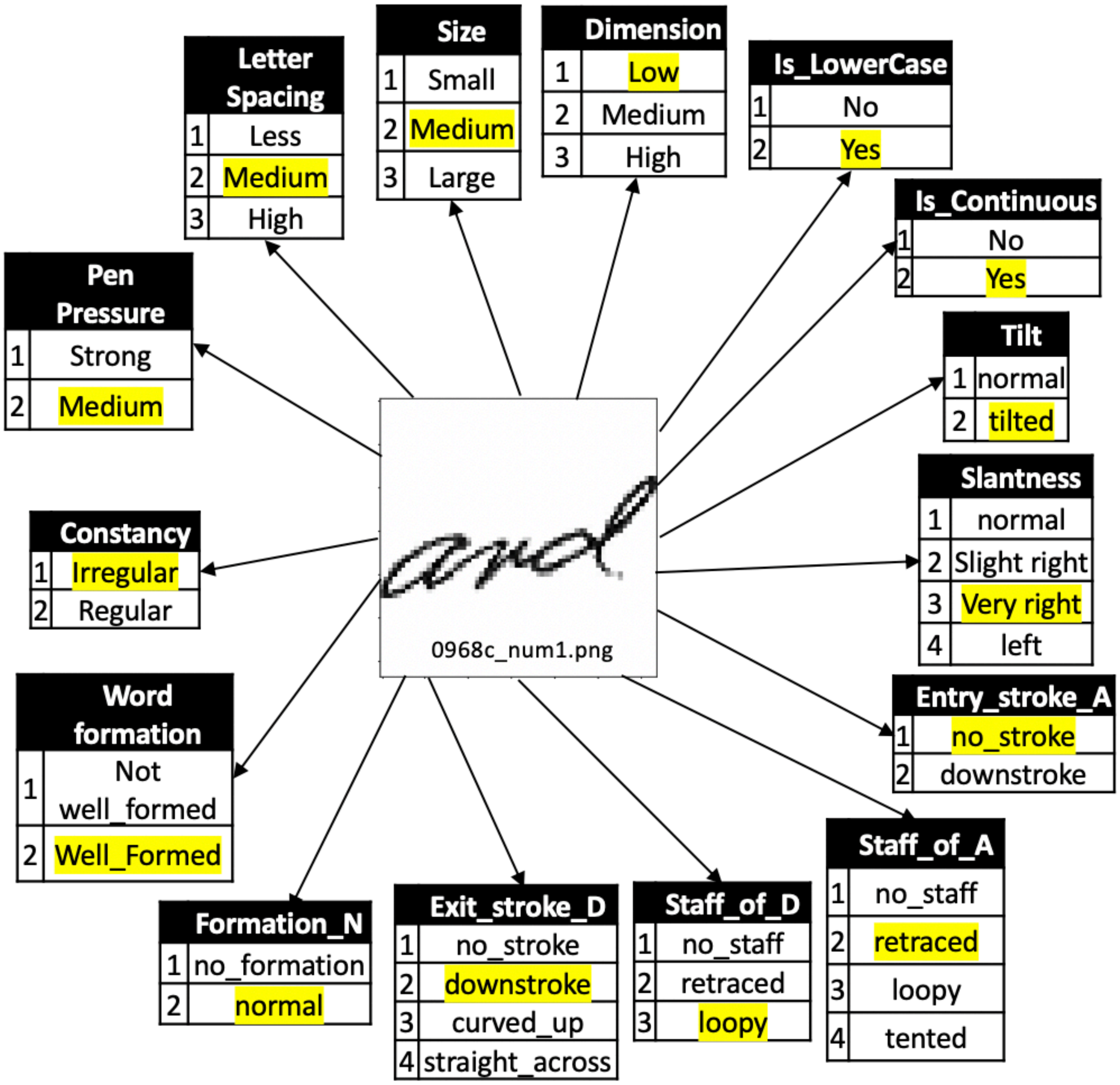
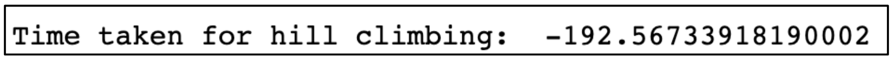


Figure 1: Human observed features mapped to an example image

Task 2: Probabilistic Approach
We create a 15 feature Bayesian model for an image with 15 nodes and estimate the dependencies using the handcrafted annotated features of all the images and performing a hill climbing search. The model for image 1 created is given in figure 2.



```
Time taken for hill climbing:  -192.56733918190002
```

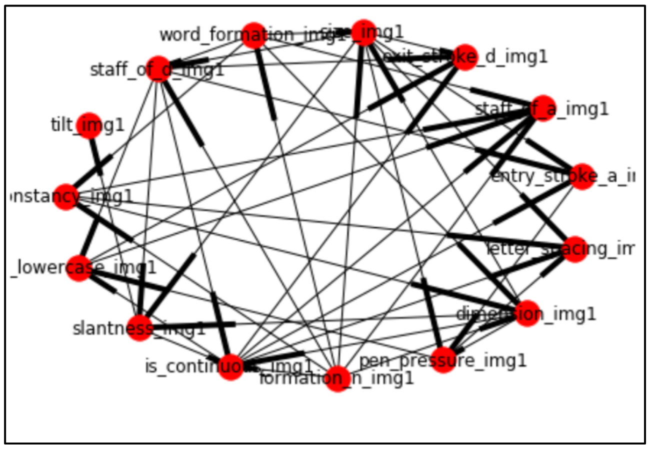


Figure 2: 15-nodes Bayesian model for image 1

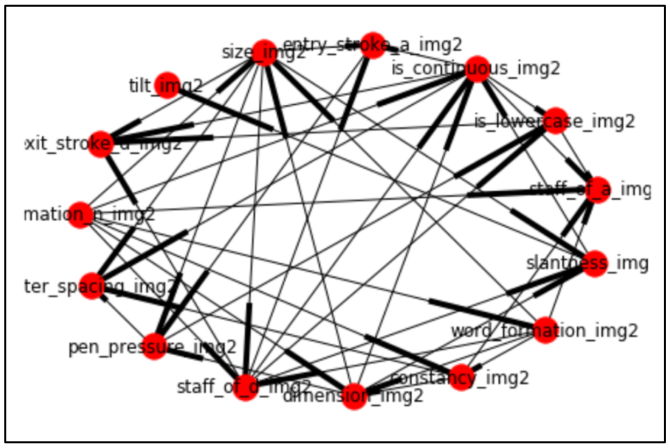


Figure 3: 15-nodes Bayesian model for image 2

We then clone this model to generate a similar model for image 2 (Figure 3). These two models are then joined using a hypothesis node which was connected to the pen pressure and letter space nodes in both the models. These dependencies were determined by examining the CPDs of all nodes manually. This resulted into a combined model (Figure 4) that takes into consideration the features of both the images and generates CPDs for the hypothesis node. The hypothesis node can have values 0 or 1 for ‘different writers and ‘same writers’ respectively.

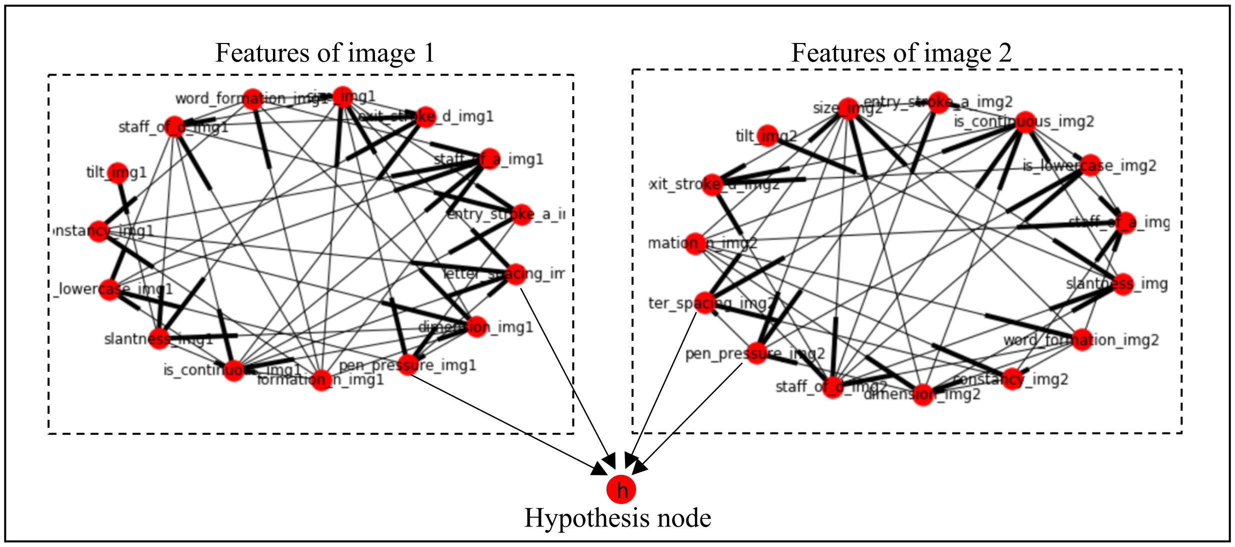
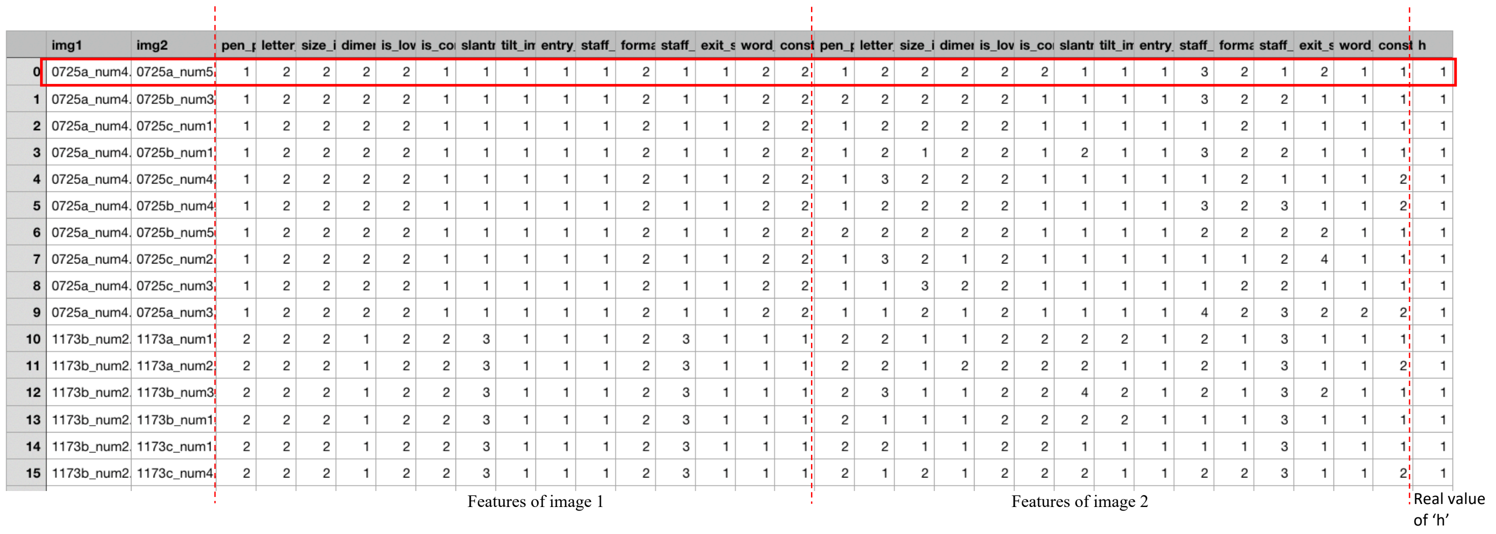


Figure 4: 31-nodes Combined Bayesian model for comparing two images

Inferences of the Bayesian model:

We use the Bayes rule to infer the value of hypothesis node given the values of all features of both images.
Bayes Rule: $P(\text{hypothesis}|\text{data}) = \dfrac{P(\text{data}|\text{hypothesis})\, P(\text{hypothesis})}{P(\text{data})}$



	img1	img2	pen_	letter	size_i	dimer	is_lov	is_co	slantr	tilt_in	entry	staff_	forma	staff_	exit_s	word	const	pen_	letter	size_i	dimer	is_lov	is_co	slantr	tilt_in	entry	staff_	forma	staff_	exit_s	word	const	h
0	0725a_num4.	0725a_num5.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	2	2	2	2	2	1	1	1	3	2	1	2	1	1	1
1	0725a_num4.	0725b_num3.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	2	2	2	2	2	1	1	1	1	3	2	2	1	1	1	1
2	0725a_num4.	0725c_num1.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	2	2	2	2	1	1	1	1	1	2	1	1	1	1	1
3	0725a_num4.	0725b_num1.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	2	1	2	2	1	2	1	1	3	2	2	1	1	1	1
4	0725a_num4.	0725c_num4.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	3	2	2	2	1	1	1	1	1	2	1	1	1	2	1
5	0725a_num4.	0725b_num4.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	2	2	2	2	1	1	1	1	3	2	3	1	1	2	1
6	0725a_num4.	0725b_num5.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	2	2	2	2	2	1	1	1	1	2	2	2	2	1	1	1
7	0725a_num4.	0725c_num2.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	3	2	1	2	1	1	1	1	1	1	2	4	1	1	1
8	0725a_num4.	0725c_num3.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	1	3	2	2	1	1	1	1	1	2	2	1	1	1	1
9	0725a_num4.	0725a_num3.	1	2	2	2	2	1	1	1	1	1	2	1	1	2	2	1	1	2	1	2	1	1	1	1	4	2	3	2	2	2	1
10	1173b_num2.	1173a_num1.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	2	1	1	2	2	2	2	1	2	1	3	1	1	1	1
11	1173b_num2.	1173a_num2.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	2	1	2	2	2	2	1	1	2	1	3	1	1	2	1
12	1173b_num2.	1173b_num3.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	3	1	1	2	2	4	2	1	2	1	3	2	1	1	1
13	1173b_num2.	1173b_num1.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	1	1	1	2	2	2	2	1	1	1	3	1	1	1	1
14	1173b_num2.	1173c_num1.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	2	1	1	2	2	1	1	1	1	1	3	1	1	1	1
15	1173b_num2.	1173c_num4.	2	2	2	1	2	2	3	1	1	1	2	3	1	1	1	2	1	2	1	2	2	2	1	1	2	2	3	1	1	2	1

Features of image 1

Features of image 2

Real value of 'h'

Consider the marked example in the above table, we infer the value of 'h' using our Bayesian model and compare it with actual label to get the accuracy.
To infer 'h', we calculate $P(h \mid pen\_pressure\_img1{=}1,\ letter\_spacing\_img1{=}2, size\_img1{=}2, \ldots\ldots, exit\_stroke\_d{=}2, word\_formation\_img2{=}1, constancy\_img2{=}1)$

The code snippet to infer the hypothesis and get the accuracy is given below:

```
i=0
counter=0
combined_features_test = combined_features_test.sample(frac=1).reset_index(drop=True) #for shuffling  the rows

start = time.time()
for index,row in combined_features_test.iterrows():
    print(i,"\n","actual h :", row['h'])
    mle1 = infer.map_query(variables={'h'},
                           evidence = {'pen_pressure_img1' : row['pen_pressure_img1'] - 1, 'letter_spacing_img1
                           #evidence will be the values of all 30 features
    print(mle1) #predicted h
    if(mle1['h']==row['h']): #if actual=predicted
        counter+=1
    i +=1

end = time.time()
print("time taken for",i, "inferences :", end - start)
print("Correct predictions : ", counter , " out of ", i)
print("accuracy:", (counter/i)*100, "%")
```

```
1971
 actual h : 0
{'h': 0}
1972
 actual h : 1
{'h': 0}
1973
 actual h : 0
{'h': 0}
time taken for 1974 inferences : 503.9578070640564
Correct predictions :  1694  out of  1974
accuracy: 85.81560283687944 %
```

Task 3: Deep Learning Approach

A Siamese Convolutional Neural Network was created where the input is 2 images and the outputs are 0 or 1 (Similar/not similar). The inputs are converted into two non-explainable feature representations with the help of 2 similar CNNs with shared weights. These features are then merged and passed onto a classifier which predicts the class.

We first create a data generator function that reshapes the images to (64,64,1) and passes the images in pair and in batches of 64 to the model. This is done so as to avoid overfitting and also to save execution memory. The data generator function used is as follows:

```
def datagen(image_triples, batch_size):
    while True:
        # loop once per epoch
        indices = np.random.randint(0,len(image_triples),batch_size)
        #print(type(image_triples))
        shuffled_triples = [image_triples[ix] for ix in indices]
        num_batches = len(shuffled_triples) // batch_size
        for bid in range(num_batches):
            # loop once per batch
            images_left, images_right, labels = [], [], []
            batch = shuffled_triples[bid * batch_size : (bid + 1) * batch_size]
            for i in range(batch_size):
                lhs, rhs, label = batch[i]
                images_left.append(load_image(lhs))
                images_right.append(load_image(rhs))
                labels.append(label)
            Xlhs = np.array(images_left)
            Xrhs = np.array(images_right)
            Y = np_utils.to_categorical(np.array(labels), num_classes=2)
            yield ([Xlhs, Xrhs], Y)
```

The entire process from generating images to creating and merging different models is illustrated below:

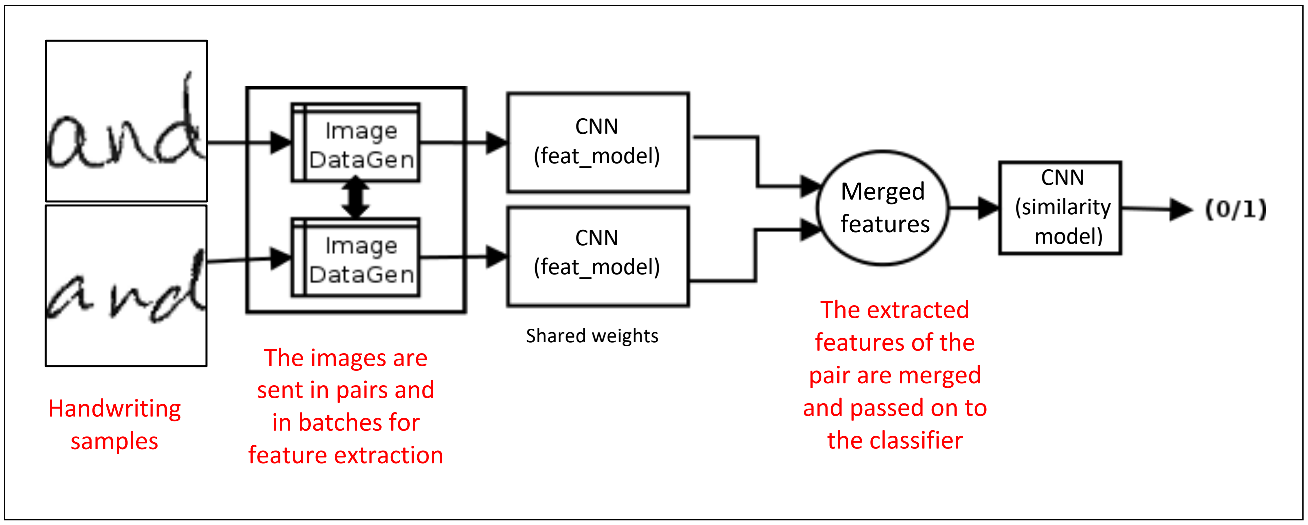


Image DataGen
Image DataGen
CNN (feat_model)
CNN (feat_model)
Shared weights
Merged features
CNN (similarity model)
(0/1)
Handwriting samples
The images are sent in pairs and in batches for feature extraction
The extracted features of the pair are merged and passed on to the classifier

Figure 5.1: Siamese CNN for handwriting verification

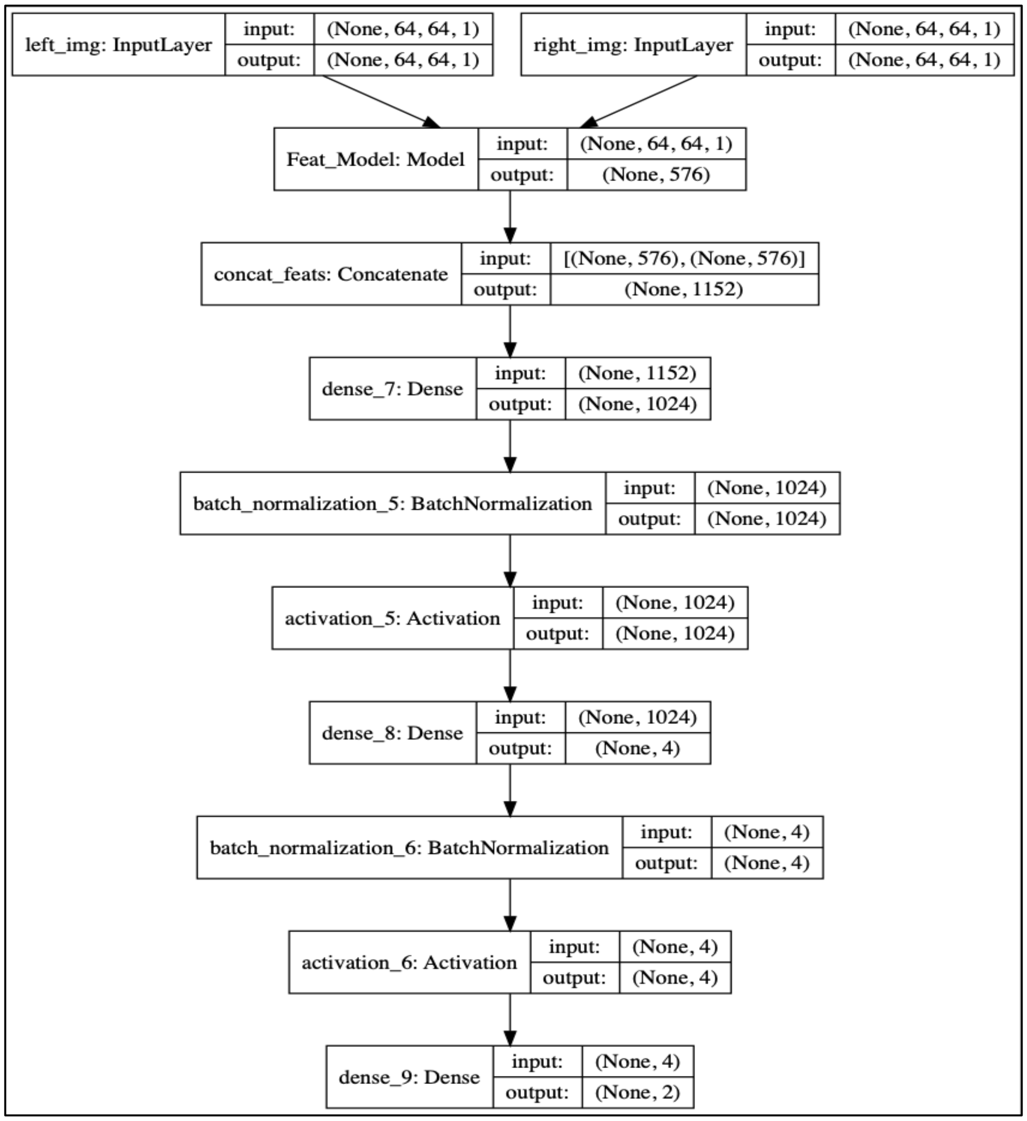


Figure 5.2: A more detailed visualization of the similarity_model

Results of the Siamese model:

After training the model for 10 epochs, the training accuracy observed was 99.92% and the validation accuracy observed was 58.71% on the seen data and almost similar on the unseen and shuffled data. The time required for each epoch was around 600 seconds. The plots for accuracy and loss for seen, unseen and shuffled data are as follows:

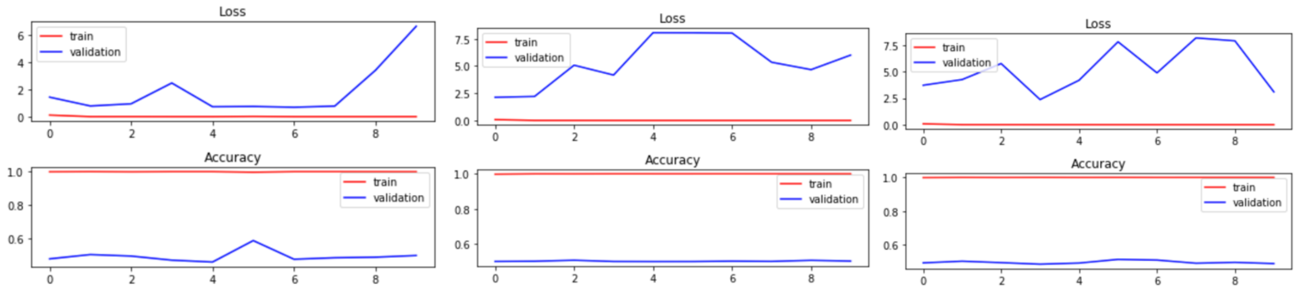


Figure 6: Accuracy and Loss plots for seen, unseen and shuffled data respectively

Task 4: Representation Learning Approach

The feature model in the above Siamese model extracts features from the given images but these features are not explainable in terms of human understandability. To make the features explainable, we pass the encoded image through 15 neural networks where each neural network represents one of the features that were identified for this project. Firstly, we train an autoencoder on a set of images so that the encoder generates correct feature representations of the images. After training the autoencoder, we connect the encoder layer to 15 neural networks. Keeping the weights of the encoder frozen, we train the neural networks on the 15 features of training images. The resulting model learns to generate an encoding of the images in terms of the required features.

1. **Encoder:**

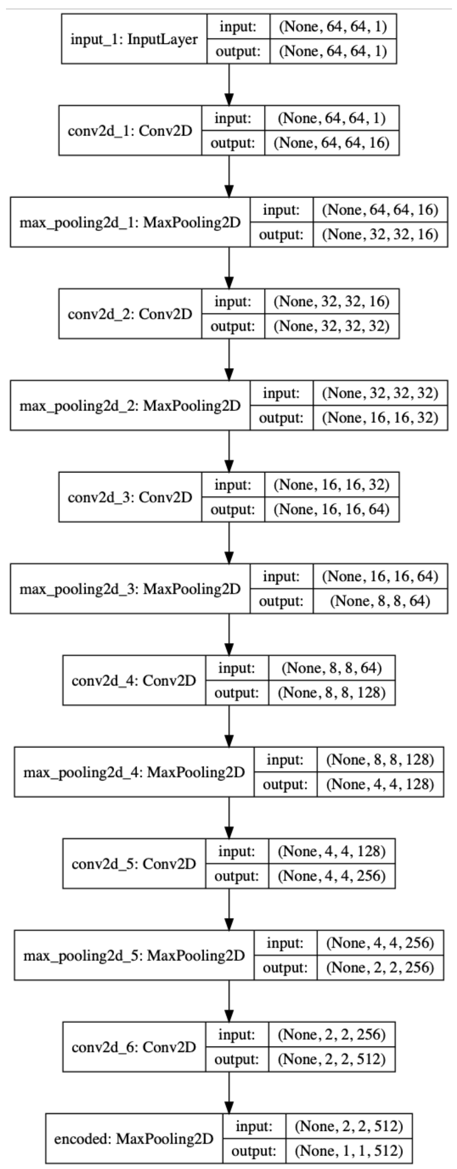

input_1: InputLayer | input: (None, 64, 64, 1) | output: (None, 64, 64, 1)
conv2d_1: Conv2D | input: (None, 64, 64, 1) | output: (None, 64, 64, 16)
max_pooling2d_1: MaxPooling2D | input: (None, 64, 64, 16) | output: (None, 32, 32, 16)
conv2d_2: Conv2D | input: (None, 32, 32, 16) | output: (None, 32, 32, 32)
max_pooling2d_2: MaxPooling2D | input: (None, 32, 32, 32) | output: (None, 16, 16, 32)
conv2d_3: Conv2D | input: (None, 16, 16, 32) | output: (None, 16, 16, 64)
max_pooling2d_3: MaxPooling2D | input: (None, 16, 16, 64) | output: (None, 8, 8, 64)
conv2d_4: Conv2D | input: (None, 8, 8, 64) | output: (None, 8, 8, 128)
max_pooling2d_4: MaxPooling2D | input: (None, 8, 8, 128) | output: (None, 4, 4, 128)
conv2d_5: Conv2D | input: (None, 4, 4, 128) | output: (None, 4, 4, 256)
max_pooling2d_5: MaxPooling2D | input: (None, 4, 4, 256) | output: (None, 2, 2, 256)
conv2d_6: Conv2D | input: (None, 2, 2, 256) | output: (None, 2, 2, 512)
encoded: MaxPooling2D | input: (None, 2, 2, 512) | output: (None, 1, 1, 512)

Figure 7.1: Visualization of the encoder layers

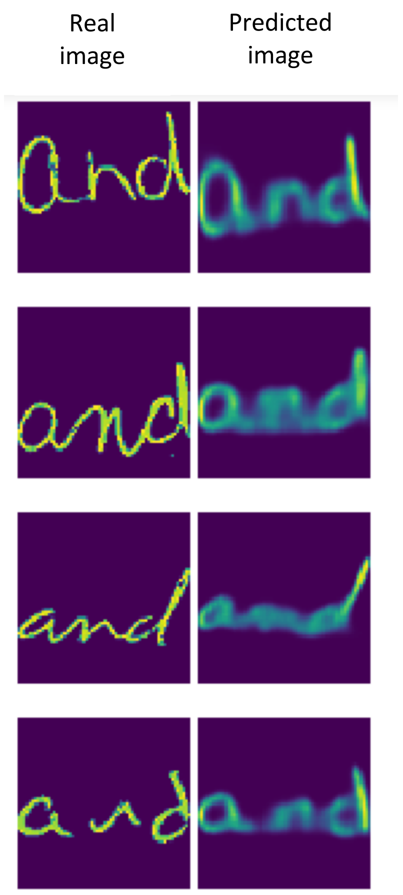
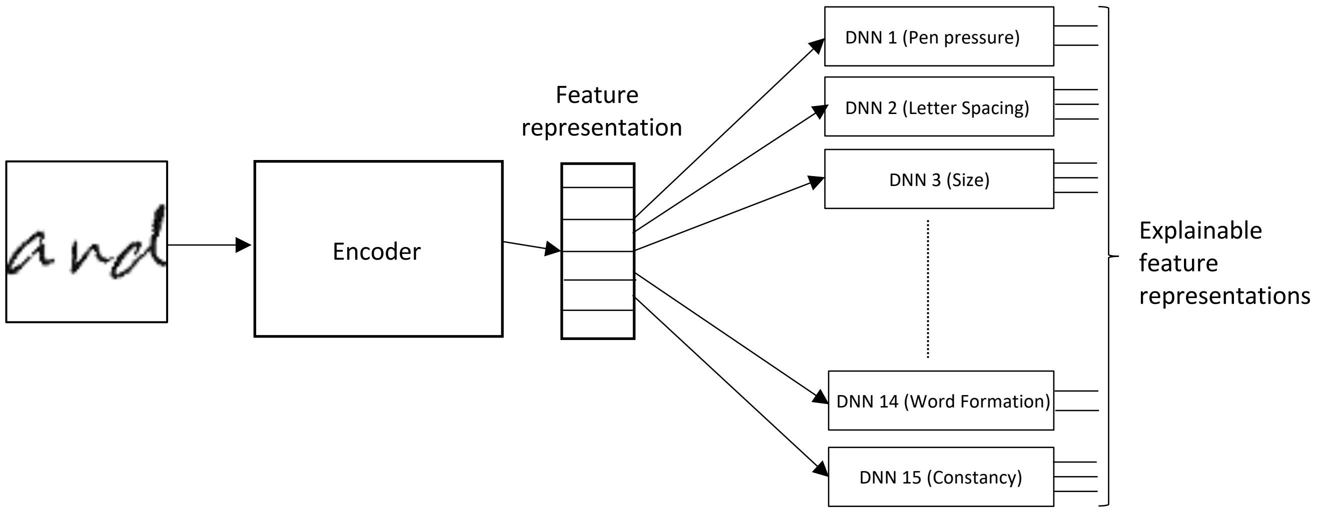


Figure 7.2: The real image vs. the reconstructed image using encoder + decoder

2. Converting features generating by the encoder to human understandable features:



15 DNN, with Dense Layers each are constructed and trained for the 15 features where each DNN represents one feature each. The outputs are the human identified feature classes. The code snippet for constructing each DNN is as follows:

```
dense_layer = encoder.get_layer('encoded').output
out_dense_layer=[]
for i in range(1,len(features_values)+1):
    out_dense_layer.append(Dense(features_values[i-1] ,
                                 activation='softmax',
                                 name = 'out_feature_'+str(i))
                                (Dense(128 , activation='relu', name = 'dense_layer_'+str(i))
                                (Flatten()((dense_layer)))))

human_features = Model(inputs=encoder.input, outputs=out_dense_layer)
human_features.summary()
```

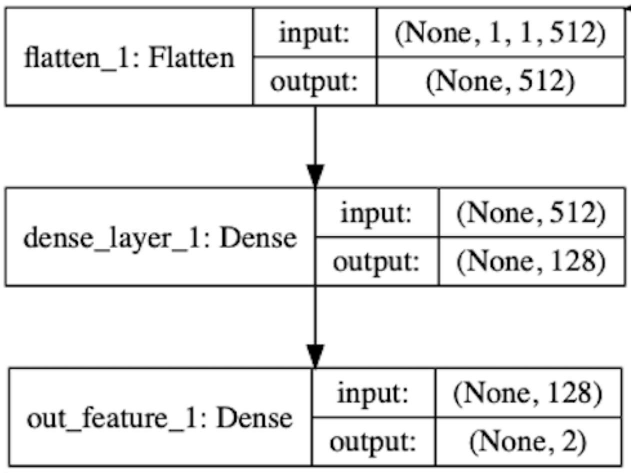
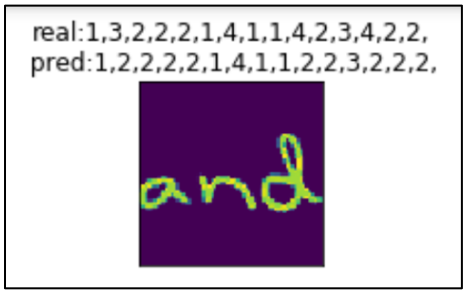


Figure 8: Visualization of DNN 1

Results of the Representation learning approach:

We calculate the feature-wise similarity score using cosine similarity function to calculate the cosine similarity of each predicted feature with the actual human annotated feature.



real:1,3,2,2,2,1,4,1,1,4,2,3,4,2,2,
pred:1,2,2,2,2,1,4,1,1,2,2,3,2,2,2,

Figure 9: The real and predicted features of a sample image

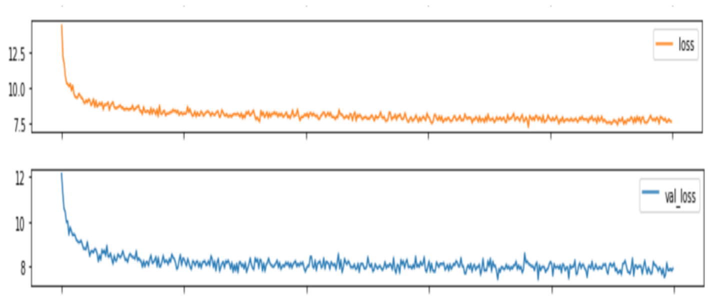


Figure 10: The plot of loss and validation loss for our representation learning model

We also calculate the overall cosine similarity between each feature of all the images and observe that the feature 'tilt'(F8) is predicted most accurately and the feature 'entry stroke of a' has the least cosine similarity between the real and predicted labels. The following table shows all the overall cosine scores between real and the predicted labels in the validation dataset.

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**
Overall cosine score	0.975	0.947	0.9702	0.956	0.975	0.910	0.960	0.992	0.880	0.975	0.976	0.926	0.925	0.916	0.949

3. Conclusion

We used the probabilistic approach and the neural approach to compare two handwritten images and observed that the Bayesian model gave a decent accuracy of 86%. Though the CNN approach gave an accuracy of 99% on the training data, it only gave around 50% accuracy on the validation data. This clearly implies that the model is overfitting on the training data even after tuning the hyperparameters. We can look into various regularization method to make the model not overfit the training data.

Finally, we tried to automate the manual annotation task by generating the features for all the images using representation learning. We then use these features to calculate the similarity between two images. We see that this approach predicts almost accurate results for features like tilt, lowercase, letter spacing but doesn't always predict correct labels for features like entry stroke of 'a'.