Offline Signature Verification using Siamese Convolutional Neural Networks

Yash Dixit, Kshitij Kapoor

1 Abstract

This report presents a Siamese model that determines whether the signature in a given image is genuine or forged, given very few images with genuine signature from the supposed signer. This siamese model uses a convolutional neural network for generating embedding of images. Rather than generating a new deep network for each signer, the model maintains a shared Convolutional Neural Network for all signers which extracts features from the input image with signature that allows model to differentiate between genuine and fake signatures. This report also presents a method for using image augmentation so that it can be used to increase the accuracy of the Siamese model which is presented in the paper. Experimental results conducted on the ICDAR 2011 dataset, show that the model generalizes well and can be used to distinguish genuine and forged signatures.

2 Introduction and Motivation

Signature verification broadly falls into two categories , namely online and offline verification. While online verification deals with capturing signatures in real time with electronic devices such as signature pads, offline signature deals with capturing images, usually 2-Dimensional, using scanning or imaging devices. Establishing authority to verify and identify an individual digitally is an actively growing area of research. The increasing role of signatures, in a digital medium, to facilitate financial transactions, document analysis, access control along with authenticating individuals on the internet is a borrowed process from the actual physical methodology of a signature being representative of an individual's identity. In a traditional framework, an assigned individual manually compares the signatures to a signature database and makes a judgement over the validity of the signature. Given the inadequate amount of working hours to complete this monotonous process, such a validation is subject to unpredictability and human error.

In a digital medium, it becomes increasingly important to ensure that any discrepancies in the process of verification are immediately recognised and rectified. By the virtue of their nature of being gathered, online signature verification is an easier task due to increased number of feature representations such as capturing the slant of the signatures, the pressure of the signature, sequences of coordination between the stylus and signature pad. In comparison, the offline signature verification task is a relatively challenging task due to the limited amount of information that it provides with. Because of the absence of inputs like pen trajectory and pressure, training a model which provides reasonable accuracy in Offline Signature Verification is harder than training a model for Online Signature Verification. In an attempt to solve this offline verification inefficiency problem and ensure reliable accuracy levels, we have implemented a model that deals with verifying human

signatures i.e. learn a set of (name, signature images) pairs and then given a name and a signature, it is able to say if the signature is authentic or a forgery.

3 Background and Previous Work

A plethora of research has been conducted in the area of signature verification in the past both using the online method as well as the offline method. We have partitioned our background research on two grounds, one that covers implementation that takes into account siamese networks and then all the others. This was done primarily to highlight how state of the art implementations have one thing in common, the usage of a siamese network implementation. Both these broad aspects include a great number of publications along with some prominent implementations of the same. We have listed and described a few of them in the sections below.

3.1 Non Siamese Models

3.1.1 Hidden Markov Model

The Hidden Markov Model (HMM) approach [8] is a probabilistic pattern matching technique which absorbs variability and the similarity between signature samples. The paper on the system outline for the given approach explains how a grid-segmentation scheme has been implemented along with an HMM to perform required tasks. Each signature image is superimposed on a grid thereby splitting the image into different segments. Each of these segments are then used to calculate the pixel density where each of these pixel densities represent a local feature. This in turn means that all these images are represented as a sequence of feature vectors with each feature vector being associated with the segment pixel density. To achieve better results, the implementation makes use of different segment sizes to analyse the signatures under multiple resolutions and scales. These vectors are then converted into discrete symbols using k-means vector quantization algorithm. The quantization algorithm aims to find encodings of these vectors that reduce the expected distortion between them. An ad hoc hidden markov process is considered to be producing a set of signatures for each author. Hence, the HMM is able to capture the local characteristics in each of these segments, outputting probabilities that help in the verification process.

Despite HMM being deployed in the field of offline signature verification for numerous years, it still faces some obvious drawbacks and has its own intrinsic limitations. The most significant of these limitations is its poor discriminative power which limits its application on the signature verification. Hence a modified version of the HMM along with ANNs is used to provide state of the art results.

3.1.2 Artificial Neural Network Approach

Many methods of image processing-based learning tasks revolve around a process of feature extraction. The CNN approach [7] uses features extracted from preprocessed signature images from 55 individuals. The implementation first talks about extracting features from the samples of images obtained. This process involves extracting a set of elements that help uniquely identify the sample belonging to a particular class. These features are broadly classified into three types: global, statistical and topological or also popularly known as the GSC (gradient, structural and concavity features). The gradient features track the change in magnitude in a 3x3 pixel window. Structural features capture patterns such as strokes, and concavity features capture topological features such

as bays and holes. The next step involves mapping the signature images to a feature space where the above mentioned features represent the signature. After this step, a typical density estimation problem is solved to find an appropriate probability density function that fits the signature samples in feature space. The classification step for the approach is to learn the relationship between the signature and its class as either genuine or a forgery. Convolutional neural networks do an excellent job of verifying signatures when allowed access during training to examples of genuine and forged signatures.

The method takes care of simple and random forgeries and skilled forgeries are also eliminated to a great extent. However, imitating a signature allows one to produce forgeries so similar to the originals that discrimination is sometimes impossible using the neural networks. It is complicated due to the large variability introduced by some signers when writing their own signatures.

3.1.3 Template Matching Approach

This approach [3] is one of the simplest approaches and it is mainly used for pattern recognition. It uses the behavioral biometrics of a hand-written signature to confirm the identity of a computer user. An individual signs on the digitizing tablet using a special pen. Regardless of the signature size and position, the signature is characterized as pen-strokes consisting of x, y coordinates and the data is stored in the signature database. The optimal matching of the signature patterns is based on the number of strokes of the two-dimensional signature patterns. Individual strokes are identified by finding the points where there is a decrease in pen tip pressure, decrease in pen velocity, and rapid change in pen angle. Given a test signature to be verified, the positional variations are compared with the statistics of the training set and a decision based on a distance measure is made.

3.1.4 Support Vector Machine

Support Vector Machine (SVM) is a kernel-based classification method that is suitable for solving machine learning features problems even in very high dimensions, and it's frequently used in pattern recognition applications. This approach [5] involves deriving unseen data by adjudging differences between classes using a high dimensional feature space of given data. A set of signature samples are collected from individuals and these signature samples are scanned in a grayscale scanner. These scanned signature images are then subjected to a number of image enhancement operations like binarization, complementation, filtering, thinning and edge detection. From these pre-processed signatures, features such as centroid, centre of gravity, calculation of number of loops, horizontal and vertical profile and normalized area are extracted and stored in a database separately. The values from the database are fed to the support vector machine which draws a hyperplane and classifies the signature into original or forged based on a particular feature value.

Limited number of writers and genuine signatures constitutes the main problem for designing a robust SVM model for signature verification. Available handwritten signature samples are often reduced resulting in the generation of an inaccurately trained model and hence the resulting classification does not always perform well. A modification is therefore required to reduce the number of misclassifications.

3.2 Siamese Networks and One Shot Learning

3.2.1 Signature Verification using Siamese Time Delay Neural Network

The implementation [1] describes an algorithm for verification of signatures written on a pen-input tablet (online method) using a novel "Siamese" time delay network approach. This model consists of two identical sub-networks joined at their outputs. The input to these networks are signatures captured through the 5990 signature capture device. The device captured hard to replicate features such as the trajectory of the signatures as well as the pressure the pen had on the signing device. The device returned the coordinates of the signatures as a function of time (800 sets of x ,y and pen up-down points). These gathered coordinates were then converted into a form that were invariant to the position of the slope and position of the signature and some normalization was performed on this data. They were then relengthed to an appropriate value to be given as inputs to the Siamese network.

These were then fed into two seperate sub-networks based on Time Delay Neural Networks. Each of these two networks had as input the preprocessed signatures. They then acted on each of the given input patterns and worked on extracting features such as the speed at each point, centripetal acceleration, tangential acceleration, x position, y position etc (a total of 8 features). This produced signature feature vectors from each of the network models and then a distance measure was calculated by finding the cosine of the angle between two feature vectors. The training on the two networks was constrained by making sure that the weights and hyperparameters for both the models are kept the same. The training on the network was carried out using a modified version of back propagation. The verification process made use of only one of the trained networks. The feature vectors for the last six signatures signed by each person were used to make a multivariate normal density model of the person's signature. The process then consisted of comparing a feature vector for a questioned signature with the probabilistic model of feature vectors from previous signature of the same signers. Depending on the likelihood of the questioned feature vector, the signature was labelled genuine or forged.

3.2.2 SigNet

In this paper[2], the authors model an offline writer independent signature verification task with a convolutional Siamese network. This consists of twin convolutional networks that share parameters which ensure that two similar images can not be mapped to different locations in the feature space. The convolutional network takes as input resized images which are normalized by dividing the pixel values with the standard deviation of the pixel values of the total images in the dataset. The CNN structure consists of multiple convolutional layers of different kernel sizes, pooling layers and dropout layers.

These twin networks are then joined by a loss function at the top. A differentiable loss function known as the contrastive loss is chosen so that the network weights can be optimized. Contrastive loss works in the following manner. Given two sample signature images to a CNN, the contrastive loss calculates the distance between a positive sample and the target sample and contrasts that with the distance between a dissimilar sample and the target sample. In simple words, it minimizes distance between similar pairs while simultaneously maximizing it between dissimilar pairs. In the given implementation, this distance is the euclidean distance. This ensures signatures images of the same class are closer to each other in their feature space while those that are dissimilar are not. The implementation then makes use of a defined threshold value of distance to classify images

as forged or genuine. It is interesting to note that in this implementation the offline method is optimized to such standards so as to give a claimed 100% testing accuracy on the CEDAR dataset, resulting in a state of the art implementation.

3.3 ICDAR 2011 Dataset

The dataset [4] that we used for our project is the ICDAR 2011 dataset that is collected from real signatories. These signature samples were gathered by the process of writing on a paper that was connected to a digitizing tablet. The dataset therefore consists of both online and offline formats of these signatures. For each of these formats, the data was divided into a training set and a testing set following their naming conventions. We made use of the offline format of this dataset for the purpose of our project. The dataset consisted of signatures, both forged and genuine, in Dutch and Chinese languages. The offline Chinese signature dataset consisted of 10 authors which provided 235 genuine signatures and 340 forged signatures in the training set. The testing set for the same consisted of 10 authors who provided 116 reference signatures, 120 questioned signatures and 367 forged signatures. On the other hand, the Dutch signature dataset consisted of 10 authors who provided 240 genuine signatures and 123 forged signatures for the training data set. The testing data set consisted of 54 authors who provided 648 reference signatures, 649 questioned (actually genuine) signatures and 638 questioned (actually forged) signatures. The signature images were all in .png format scanned at 400 dpi and had RGB color representations.

4 Approach

After reviewing multiple state of the art implementations and carefully examining each of their advantages and disadvantages, we decided that we will use a Siamese network which uses convolutional layers and fully connected layers for verifying signatures in offline mode. The main advantage of a siamese network is that it allows for one shot learning. One shot learning approach allows the same network to be trained on signatures from multiple signers. Similarly, the same network will be used while inferring in production to determine whether the signature is forged or genuine. Using a siamese network allowed us to attain satisfactory accuracy in offline signature verification, while training on a very small dataset which consisted of data from only 10 signers. The accuracy was measured on a separate dataset which did not have any signers in common with the training dataset.

Firstly the concept of an embedding network needs to be discussed. The embedding network is a major component of a siamese network. The embedding network is a simple feedforward network. The input to this network is an image. The image can be grayscale in which case each pixel in the image would correspond to one unsigned integer of 1 byte. So, for example if the size of the grayscale image is $B \times H$, then the dimensions of the image array would be $B \times H \times 1$. If the image is colored then each pixel in the image would be represented by three unsigned integers of 8 byte each. Hence, the dimensions of a $B \times H$ colored image would be $B \times H \times 3$. The first few layers of an embedding network would be convolutional layers and pooling layers. After a few convolutional layers and pooling layers, the output from these networks is flattened and this flattened vectors acts as an input to a dense fully connected network which is a few layers deep. Appropriate activations functions like ReLU, sigmoid and tanh are used to introduce non linearity in the network. The last dense layer in the embedding network has the sigmoid activation function. This network, given an image, returns an embedding which is a flat vector, henceforth referred to as the embedding.

To train this embedding network, a custom loss function called the triplet loss [6] is required.

The intuition behind the triplet loss function is that it tries to update the weight of the embedding network in a way, so that a selected distance metric between the embedding of two genuine signatures of the same signer is minimized whereas the same distance metric between the embedding of a genuine signatures of the same signer and the embedding of a skilled forgery is maximized. Both of these tasks are done in conjunction. This forces the embedding network to learn features that can effectively distinguish a forged signature from a genuine one given the name of the actual singer is known.

A dense network, henceforth referred to as the comparator, was also used. The comparator is a fully connected dense network. The input to the comparator is a flat vector. The element wise difference vector between the embedding of a known genuine signature and the embedding of the signature in question is fed into the comparator and the output is a single value between 0 and 1. Values closer to 0 indicate that the questioned signature is forged whereas values closer to 1 indicate the questioned signature is genuine. The comparator allows the model to go from embeddings of images with questioned signatures to a genuine or a forged label.

While training and inferring from the model, image augmentation is used to introduce small modifications like rotating the image, horizontal shift, vertical shift and zoom. This allows the network to generalize well on a very small dataset. Also this makes sure that the model would not suffer when a slanted signature is fed into the network. The horizontal and the vertical shifts account for cases in production where the scanning process was not done properly and only a part of the signature was captured. The zoom allows the model to account for varying size which might change from one signature to another.

We believe that the use of this siamese structure and data augmentation would allow our model to perform well on unknown signatures. More details about how these two networks were trained and are used while inferring will be discussed in the next section.

5 Implementation Details

5.1 Siamese Model

5.1.1 Embedding Network

The embedding network in a siamese model, is simply a feedforward network which takes an image as the input and outputs a flat vector. The input to the model is a tensor of the shape $B \times H \times C$ where is C is the number of color channels in the image input. This input is fed into a combination of convolutional layers and pooling layers. Dropout is also used to make sure that the model generalizes well and does not overfit. The first part of the embedding network is just a combination of the layers stated above. The output from this part is flattened and fed into a fully connected network with multiple dense layers. This model also has dropout layers in between dense layers to make sure that the model generalizes well and does not overfit. If the last dense layer in the fully connected part has e neurons, then the output from the embedding network is a vector in the space R^e. To make sure the network can generate non linear features, ReLU is used as the activation function in all layers except the last dense layer. The last dense layer has a sigmoid activation function. Hence the embedding network E is a non linear transformation which maps an input image tensor I of the shape $B \times H \times C$ into a flat vector V of the form R^e. Hence, $E(I) = V$

In the specific implementation, which lead to best results and with which we calculated the empirical results, the input image tensor is of the shape $64x64x3$ and $e = 64$, that is the output vector V is of the form R^{64}. The specific topology used for the embedding network is in the table below.

Layer Type	Parameters
Conv2D	64 Filters of Size (3,3) with stride (1,1), ReLU Activation Function
MaxPooling2D	Pool Size (2,2)
Dropout	$\alpha = 0.25$
Conv2D	64 Filters of Size (3,3) with stride (1,1), ReLU Activation Function
MaxPooling2D	Pool Size (2,2)
Dropout	$\alpha = 0.25$
Conv2D	128 Filters of Size (3,3) with stride (1,1), ReLU Activation Function
MaxPooling2D	Pool Size (2,2)
Dropout	$\alpha = 0.25$
Flatten	
Dense	256 Neurons, ReLU Activation Function
Dropout	$\alpha = 0.25$
Dense	256 Neurons, ReLU Activation Function
Dropout	$\alpha = 0.25$
Dense	64 Neurons, Sigmoid Activation Function

Specific Topology of the embedding network used for Empirical Results

5.1.2 Triplet Loss

Triplet loss function, is used in this model, to optimize the network in a way so that for a given signer, embeddings of images of genuine signatures is closer than embedding of an image of a genuine signature and the embedding of an image of forged signature. The embedding network needs to be optimized in such a way that both these goals are achieved concurrently. First an anchor image I_A is chosen which has a known genuine signature. Next another image with a known genuine signature is chosen and is referred to as the positive image I_P. Finally an image with a known skilled forgery is chosen and is referred to as the negative image I_N. It should be reiterated that all these images are for the same signer. The embedding of these images is calculated using the embedding network. To further define this, a distance metric D is chosen. For example, the distance metric can be the L_1 distance, the L_2 distance or even the square of the L_2 distance. The triplet loss function is defined below

$$J(I_A, I_P, I_N) = max((D(E(I_A), E(I_P)) - D(E(I_A), E(I_N)) + \alpha), 0)$$

The presence of the α term makes sure that the embedding network does not converge to the trivial solution in which the embedding network outputs the same embedding for all inputs and hence if the α was absent or set to $\alpha = 0$, the loss would also be 0. Hence, this specific loss function allows the network to converge in such a way that embedding for two images with genuine signatures of a signer move closer and embedding for an image with a genuine signature and embedding for an image with a skilled forgery for the same signature move further apart in the space R^e

We experimented with all the distance metrics mentioned above and after some hyper parameter optimization, for the specific implementation, with which we calculated the empirical results, we used the square of the L_2 distance as the distance metric D and set the value of $\alpha = 1$. Hence

the loss function was

$$J(I_A, I_P, I_N) = max(((||E(I_A) - E(I_P)||_2)^2 - (||E(I_A) - E(I_N)||_2)^2 + 1), 0)$$

5.1.3 Triplet Generation

The anchor image I_A which has a genuine signature, the positive image I_P which also has a genuine signature and the negative image I_N which has forged signature of the same signer, together are called a triplet. To optimize the Siamese network so as to minimize the triplet loss function, a dataset consisting of these triplets is required. The ICDAR training dataset consists of data for only 10 signers, each with a unique ID and 24-25 images of genuine signatures and 12-13 images of skilled forgeries.

To construct a batch of the triplet dataset, we randomly sample with replacement a sample of size equal to the desired batch size from the set of IDs. For each ID in this sample, we sample one image from the set of genuine signature images, another image from the set of genuine signature images and finally one image from the set of skilled forgery signature images. The first image is the anchor, the second the positive and the third the negative. Hence for each ID in the sample a new triplet is created. Even if the same ID is present in a batch multiple times (which it will be because our batch size is greater than number of IDs in the training set), the chosen anchor, positive or negative might be different.

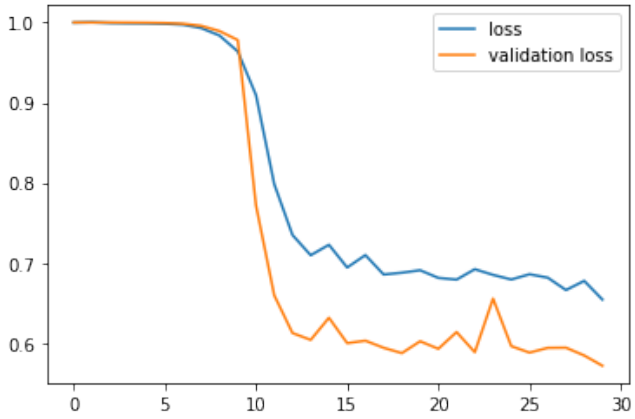
5.1.4 Image Augmentation

While inferring, it is possible that the questioned image has a signature which is tilted, or because of improper scanning, is not at the center of the scan, or a part of it has not been captured in the scan, or is a little bigger or smaller than the previous authenticated signatures we have for the person. To take care of these possibilities we used the concept of stochastic image augmentation. Image augmentation adds random tilts, vertical shift, horizontal shifts and random zoom. This image augmentation step is supposed to simulate the effects of shifted and possibly partially captured signatures due to faulty scanning, tilted signatures and size variant signatures. We believe this step would help the model generalize better while inferring in production. We apply the data augmentation method to each image in each triplet. There is no correlation between the tilting, shifting or zoom that is applied to an anchor in a triplet to that which is applied to a positive or a negative in the same triplet. Hence, all these stochastic transforms are applied independently to each image.

5.1.5 Training and Dataset Generation

The Siamese model with the triplet loss described in the sections above was implemented in the TensorFlow framework. After some experimentation we decided to use the AdaGrad optimizer. We also experimented with RMSProp and Adam but the results were not as good. Before we began the training process, we used the triplet generation and image augmentation part to generate a validation set. This might appear a little concerning because the accepted practice is to separate the validation set from the training set but we would like to point out, that on each epoch a new random batch of triplets was being generated, randomly augmented and then used for optimization of the network and hence the probability of the network training and validating on the exact same dataset for 30 epoch is very small. For some of our network configurations, the loss on the training function was decreasing quickly but the validation loss was not decreasing which made us realize

that we might be over fitting. After a lot of hyperparameter optimization we settled on using AdaGrad with a learning rate of 0.001. We trained the network for 30 epochs with 500 steps in each batch. We saved the embedding network weights for which we got the least validation loss.



Blue - Training Epoch Batch Loss, Yellow - Validation Sample Loss, Computed with $\alpha = 1$

5.2 Comparator

The job of the comparator is to determine whether the signature in the questioned image is forged or genuine given a sample of a genuine signature for the same signer

5.2.1 Input and Output

The input to the comparator network is the element wise difference of embedding of questioned image from the embedding of a genuine image and hence it is a flat vector in the space R^e. More precisely, Comparator Input $C_I = E(I_G) - E(I_Q)$, where I_G is an image with the genuine signature and I_Q is an image with the questioned signature. The output for the comparator is a value between 0 and 1. A value closer to 0 signifies that the questioned image had a genuine signature and a value closer to 1 signifies the questioned image had a forged signature. A binary crossentropy loss function is used for optimizing the network. Hence the comparator is simply a non linear map $C : R^e \to [0, 1]$

5.2.2 Network Topology and Loss Function

The comparator is a dense fully connected network with a few hidden dense layers and dropout layers. The final dense layer has just one neuron and a sigmoid activation function. All the hidden dense layers have a tanh activation function to make sure the network can extract non linear features. The specific topology used for the comparator network is in the table below.

Layer Type	Parameters
Dense	1024 Neurons, Tanh Activation Function
Dropout	$\alpha = 0.25$
Dense	512 Neurons, Tanh Activation Function
Dropout	$\alpha = 0.15$
Dense	128 Neurons, Tanh Activation Function
Dense	1 Neurons, Sigmoid Activation Function

Specific Topology of the comparator network used for Empirical Results

5.2.3 Training and Dataset Generation

We use the triplet generation method and the image augmentation method to generate dataset for training the comparator network. From each triplet, consisting of the anchor I_A, the positive I_P and the negative I_N, we generated two inputs. The first comparator input $C_I = E(I_A) - E(I_P)$ had a corresponding label 1 and the second comparator input $C_I = E(I_A) - E(I_N)$ had a corresponding label 0. All the images were augmeneted before the embedding layer was applied on them. Hence, the comparator could also account for the imperfections introduced by tilted, shifted, partial and size varying signatures.

The comparator network was trained for 100 epochs, with each epoch consisting of 100 steps. We used RMSProp with a learning rate of 0.001 and 0 momentum for optimizing the comparator.

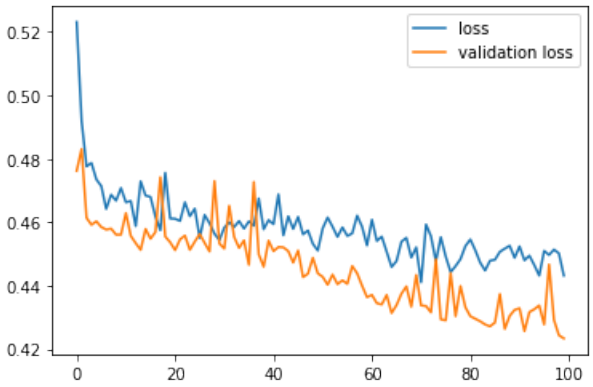


Figure 1: Loss function for the comparator over epochs

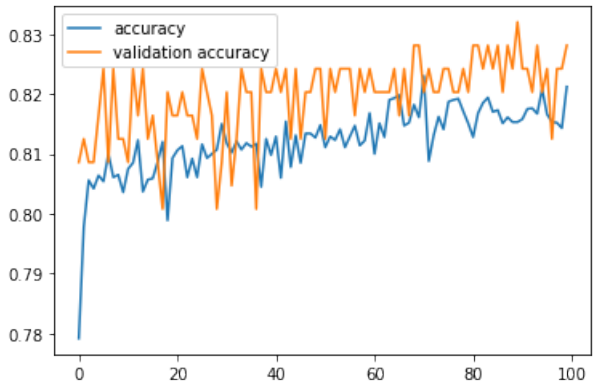


Figure 2: Accuracy for the comparator over epochs

5.3 Inference

In the previous section, we described the training procedure of all the components required for inference in real time. Now we will go over the inference process.

5.3.1 Database Generation

The ICDAR Dataset provides 12-13 images with genuine signatures and 25-26 images with questioned signatures (of which half are genuine and half are forged) for each signer. The set of signers in the training dataset and the set of signers in testing datatset are disjoint. No signer and their data is shared between the training data and the testing data. In the testing dataset, we are provided with data for 54 signers. We would like to point out that the number of signers is much higher in the testing set compared to the training set. Hence, if the model performs well on the training set, we can be sure that it generalizes well.

For each signer with a unique ID, we load all the genuine images that have been provided in the testing dataset. For all signers we have 12-13 such images. We sample with replacement 100 images from these 12-13 images. Each of the 100 images in this sample is augmented independently to produce images with slight tilt, vertical or horizontal shift and zoom. So, if there are more than 2 instances of a genuine image for a signer in this sample, it is very likely that the stochastic augmentation process would introduce some difference in the augmented images. For each augmented

image with genuine signature I_{AG}, the embedding vector $E(I_{AG})$ is calculated and stored. So for each signer, we have 100 such embedding vectors.

5.3.2 Composite Prediction

When a new questioned image I_Q is presented along with the ID of the signer, the model first computes an embedding $E_Q = E(I_Q)$ and then loads all the 100 embedding for the signer that were obtained in the database generation step. For each of the 100 embedding for the signer, we calculate the comparator input by subtracting the embedding for the questioned image from each of the 100 stored embeddings seperately. Hence the comparator input for a stored embedding E_G is $C_I = E_G - E_Q$. 100 such comparator inputs are obtained and fed into the comparator. If, out of the 100 comparator outputs, more than 50 are greater than 0.5, the model assigns a genuine tag and if less than 50 are greater than 0.5, the model assigns a forged tag to the presented image.

6 Empirical Results

Firstly, we need to go over the metrics that are used to measure the performance of a signature verification system. The first and most obvious one is the overall accuracy which is simply the percentage of signatures labelled correctly by the system. The other two important metrics are False Acceptance Rate and False Rejection Rate.

$$\text{Accuracy} = \frac{\text{Correctly Labelled Samples}}{\text{Sample Size}}$$

$$\text{FAR} = \frac{\text{No of Forged Signature Labelled Genuine}}{\text{No of Forged Signature}}$$

$$\text{FRR} = \frac{\text{No of Genuine Signature Labelled Forged}}{\text{No of Genuine Signature}}$$

It is obvious that higher values of accuracy and lower values of FAR and FRR are desired. It should also be stated that in production settings, having a very low FAR is much more important than having low FRR because a erroneously authenticated signature can lead to financial losses whereas a erroneously rejected signature might just require a bit of human intervention.

To measure the effectiveness of our model, we measure these metrics for multiple iterations on subsets of varying size of the testing set. We measure the metrics on a subset of size 10, 20, 30, 40 and 54. For each size of subset, we run 5 iterations and hence calculate 5 values of each metric. For each iteration, the subset is randomly chosen. This means that for subset of size 10, the signers that were chosen for first iteration might not be chosen in second iteration.

sample size	accuracy_mean	accuracy_min	accuracy_50%	accuracy_max
10	0.943512	0.90873	0.949153	0.979508
20	0.94113	0.912951	0.94375	0.968944
30	0.939887	0.927454	0.945225	0.951636
40	0.934857	0.926471	0.932418	0.944622
54	0.936286	0.928516	0.936286	0.947164

Accuracy score values over 5 iterations on each sample size

sample size	far_mean	far_min	far_50%	far_max
10	0.0834987	0.0241935	0.0948276	0.111111
20	0.089635	0.037037	0.1	0.133929
30	0.0839509	0.0699708	0.0743802	0.116992
40	0.0950212	0.0772443	0.0974576	0.108333
54	0.0926448	0.0782473	0.0954617	0.106416

FAR values over 5 iterations on each sample size

sample size	frr_mean	frr_min	frr_50%	frr_max
10	0.03	0.00833333	0.0166667	0.075
20	0.0291667	0.0125	0.025	0.0625
30	0.0366667	0.0277778	0.0305556	0.0611111
40	0.0358333	0.025	0.0333333	0.05
54	0.0351852	0.0277778	0.037037	0.0401235

FRR values over 5 iterations on each sample size

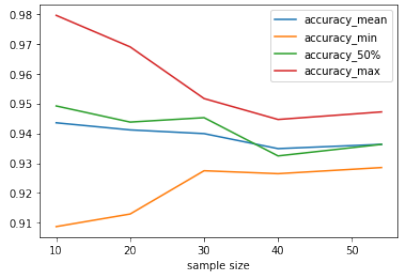


Figure 3: Accuracy Mean, Median, Max and Min

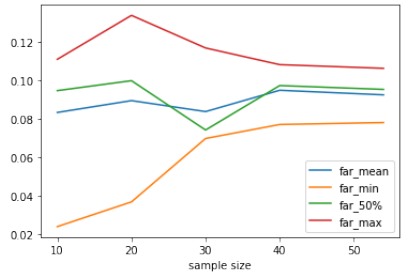


Figure 4: FAR Mean, Median, Max and Min

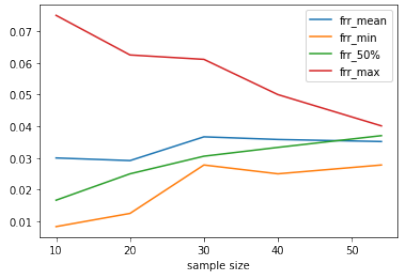


Figure 5: FRR Mean, Median, Max and Min

From the images and the table above, it is clear that the accuracy metric is pretty stable around the 93% mark for the model and so are the FAR and FRR rates. We would like to point out that our model outperformed many state of the art offline signature verification models, except for SigNet. Although SigNet claims to have 100% accuracy, that number was calculated on the CEDAR datatset. SigNet was not evaluated on the ICDAR 2011 dataset which we use in our training and testing process. Hence, we cannot be sure if it would have 100% accuracy on the ICDAR 2011 dataset. Also, on some of the other datasets, SigNet has accuracy worse than ours. But again, it is not a fair comparison.

7 Result Discussion

After analyzing the results and comparing them to state of the art implementations we were able to safely conclude that our model did indeed perform to our expectations. When performing the offline signature verification task, it is extremely important that high accuracy levels are reached so as to be able to deploy the solution in real life scenarios such as the banking sector. Therefore it was of paramount importance that we, from the beginning, targeted high levels of accuracy rates. Our Siamese network produced results that are almost parallel to the accuracies of the models that have been mentioned previously in this paper. The only difference that occurred was that varying

datasets had slightly different accuracies and some of the models were trained for hours using high-end GPUs, both of which were resources that were not available at the time of implementation for this project. We are a bit concerned about the FAR numbers that our model produced. Even then, the combination of Siamese network and comparator model produced commendable results. We believe that the Data Augmentation step for training and inferring was also responsible for increasing the accuracy metrics. The overall accuracy of our model to classify a signature as a forged or authentic over-took multiple model implementations that we have mentioned above. With most of these models only hovering around 90-91% accuracy, with the exception of SigNET, our model constantly attained accuracy levels close to 93%. It was anticipated that our model would replicate the accuracy levels of these previous implementations however it is highly commendable that our implementation over achieved our expectations.

8 Conclusion

Through the course of this project, we have presented a framework that makes use of the Siamese Neural Network for offline signature verification process. Offline signature verification makes use of signature images gathered using imaging devices and hence allows for a low barrier to entry. The methods used in this project are state of the art approaches and the results obtained using them are highly commendable. Our thorough examination of the model emphasizes the efficiency of the model structure, especially due to our rigorous testing phase with various hyperparameters and calculation methods. Our model quite efficiently classifies the various signature images as fraudulent or authentic, while processing images with different styles and handwriting backgrounds. We believe that the use of stochastic data augmentation along with sampling with replacement from the genuine image database, further enriches the learning curve. It also opens the door for exploring more possibilities in the future in terms of further developing our work into meaningful publication.

References

[1] Jane Bromley et al. "Signature Verification Using a "Siamese" Time Delay Neural Network". In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. NIPS'93. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1993, pp. 737–744.

[2] Sounak Dey et al. *SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification*. 2017. arXiv: 1707.02131 [cs.CV].

[3] A Julita et al. "Online signature verification system". In: *2009 5th International Colloquium on Signal Processing & Its Applications*. IEEE. 2009, pp. 8–12.

[4] M. Liwicki et al. "Signature Verification Competition for Online and Offline Skilled Forgeries (SigComp2011)". In: *2011 International Conference on Document Analysis and Recognition*. 2011, pp. 1480–1484. DOI: 10.1109/ICDAR.2011.294.

[5] Luis E Martinez et al. "Parameterization of a forgery handwritten signature verification system using SVM". In: *38th Annual 2004 International Carnahan Conference on Security Technology, 2004*. IEEE. 2004, pp. 193–196.

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *CoRR* abs/1503.03832 (2015). arXiv: 1503.03832. URL: http://arxiv.org/abs/1503.03832.

[7] Harish Srinivasan, Sargur N Srihari, and M Beal. "Signature verification using kolmogorov-smirnov statistic". In: *Proc. International Graphonomics Society Conference (IGS)*. 2005, pp. 152–156.

[8] A El-Yacoubi et al. "Off-line signature verification using HMMs and cross-validation". In: *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No. 00TH8501)*. Vol. 2. IEEE. 2000, pp. 859–868.