Off-line Signature Verification

Report submitted in fulfillment of the requirements
for the Exploratory Project of

Fourth Semester BTech. (Part 2)

by

Manik Goyal, Harshit Mehrotra, Vandit Jain

15075029-15075019-15075055

Under the guidance of

Prof. Rajeev Srivastava

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2017

Dedicated to

Our parents and teachers without whom this wouldn't have been possible

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi

Date:

Manik Goyal, Harshit Mehrotra, Vandit Jain
15075029-15075019-15075055
Sophomores
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Off-line Signature Verification**" being submitted by **Manik Goyal, Harshit Mehrotra, Vandit Jain** (**Roll No. 15075029-15075019-15075055**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date:

Prof. Rajeev Srivastava
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to express our sincere gratitude to **Prof. Rajeev Srivastava** for his guidance and constant supervision. He provided us with necessary information and supported us throughout the project.

Place: IIT (BHU) Varanasi

Date: **Manik Goyal, Harshit Mehrotra, Vandit Jain**

Contents

List of Figures **vii**

List of Tables **viii**

List of Symbols **ix**

Abstract **xi**

1 Introduction **1**

1.1 Overview 1

1.2 Motivation Behind the Research Work 3

1.3 Organisation of the Report 3

2 Literature Review **5**

2.1 DRT and HMM 6

2.2 DCT-SVM 7

2.3 Machine Learning for Signature Verification 7

2.4 Persian Signature Verification using CNN 8

2.5 Offline Signature Verification with CNN 8

2.6 Writer Independent Feature Learning 9

3 Project Work **10**

3.1 Pre-processing 10

3.2 Feature Extraction 13
3.2.1 Discrete Cosine transform 14
3.2.2 Convolutional Neural Network 15
3.3 Formation of dissimilarity distributions 18
3.4 Comparison of dissimilarity distributions 20
3.5 Classification 21

4 Experimentation and Results **23**
4.1 Pre-processsing 24
4.2 Feature Extraction 24
4.2.1 Discrete Cosine Transform (DCT) 24
4.2.2 Convolutional Neural Network (CNN) 25
4.3 Dissimilarity Distributions 26
4.4 Classification 28
4.5 Final Results 31

5 Conclusions and Discussion **34**

Bibliography **35**

List of Figures

3.1 Block diagram of the procedure 11

3.2 Resized grayscale image 11

3.3 Image after binarisation 12

3.4 Image after adding margins 12

3.5 Image after Morphological filter has been applied 13

4.1 Epochs-features plots for dataset A 26

4.2 Epochs-features plots for dataset B 27

4.3 Epochs-features plots for dataset C 27

4.4 ROC plot for DCT features 29

4.5 ROC plot for CNN-extracted features 30

4.6 Final results after incorporating β 33

List of Tables

4.1 Comparison of results for DCT and CNN-extracted features 31

4.2 Final performance results after incorporating β 32

List of Symbols

Symbol	Description
$\boldsymbol{J}$	Within Known dissimilarity distribution
$\boldsymbol{K}$	Questioned vs known dissimilarity distribution
N_e	Effective number of data points

Abstract

Offline signature verification is the task of determining whether a foreign signature sample indeed belongs to a person (genuine) or not (forgery). In this project, we describe a technique to achieve this verification task which is based on similarity amongst the ensemble of genuine samples of an individual. The method is applied for special learning rather than general learning, implying that learning is based on genuine samples available for a particular person because in real life applications, one does not have access to forgeries for training a verification system. General learning, on the other hand would have provided for person independent verification.

The aim of the project is to achieve successful verification based on training on a few examples. In practice, it is imperative that a new user be asked for only a few genuine samples of his/her signature.

The proposed technique is composed of three parts – feature extraction (using DCT, CNN), constructing dissimilarity distributions (using distance function) and comparing within known and known vs. questioned distributions (using Kolmogorov – Smirnov test). The result of the last part is used as a parameter for judging whether the questioned sample is genuine or forged.

Keywords: Discrete Cosine Transform, Convolutional Neural Networks, Distance function, Kolmogorov – Smirnov test

Chapter 1

Introduction

1.1 Overview

Biometric recognition, in general, forms a strong link between a person and his/her identity as biometric traits cannot be easily shared, lost, or duplicated. Handwritten signature, in particular, happens to be the most natural behavioural biometric which generally establishes a mode of providing a personal identity for authentication. The most common task in the field of forensic document analysis is that of authenticating signatures.

A signature is classified into two types; as on-line signature and off-line signature based on the procedure of capturing them. An on-line signature is acquired at the time of its registration and provides intrinsic dynamic details, viz. velocity, acceleration, direction of pen movement, pressure applied and forces. An off-line signature on the other hand works using a static image of the signature captured after the signing process is complete.

The major threat/challenge in signature verification is the forgery of the signature, which in turn can be of three types: **random forgery**, when the forger has the

access neither to the genuine signature nor to any information about the author's name; **simple forgery**, where the forger has no access to the sample signature but knows the name of the signer; and lastly, **skilled forgery**, when forger reproduces the signature, having access to the sample genuine signature. The signature verification problem is to classify a sample of signatures as genuine and forgery.[1]

When it comes to building a verification system, one thinks of training it. However, practical applications pose a limitation to the problem of training. One does not have access to forgeries of a person's signature when a generic system has to be extended to verify his signature. Moreover, such a new individual will only be able to give a few genuine samples and that is all the information that the eventual system has in order to implement verification.

A verification system can, in turn be of two types - one that performs **general learning** and another that performs **specific learning**. The former one performs verification independent of the person the signature belongs to. The latter on the other hand verifies the signature of the person whose samples it has been trained on. In this work, we have built a system that performs person dependent verification based on genuine samples of the person's signature it is fed with.[1]

Keeping in mind the constraints posed by signature verification problem, we propose a solution using Convolutional Neural Networks (CNN) used for getting features from scanned images of signatures after some preprocessing has been done. Feature vectors are fed into Kolmogorov-Smirnov (KS) Test and the result is used to classify image samples as genuine or forged.

1.2 Motivation Behind the Research Work

Signature verification is an important task in forensic document analysis. Also verification of signatures manually requires one to be an expert in the field in order to ensure low error rates. Also two people verifying the same signature can have different opinions leading to ambiguity.

Automation of this process requires a signature verification system that can classify signatures at a healthy pace along with maintaining low error rates. Also important is the problem to build a system that can verify signatures using less amount of input data. This is because in real world scenario, we get very few samples of genuine signatures to train our model upon. Such situations often occur in fields of banking, finance etc.

Off-line Signature Verification is an open problem that poses challenge for the researchers. Motivation behind the research work is, thus, to come up with a technique that can classify signatures using very few samples with the lowest possible error rates; to build a system that can solve problem in real world scenarios.

1.3 Organisation of the Report

The rest of this report has been organised as follows:

1. Chapter-2: **Literature review** describes the work that has been done in the field of signature verification by other researchers. Various methods previously used have been explained along with their results mentioned comprehensively.

2. Chapter-3: **Project Work** step by step goes through each step that we propose in our method. Each section of the chapter describes a step in detail giving mathematical expressions used.

3. Chapter-4: **Experimentation and Results** describes in detail about the steps(described in Chapter-2) performed on various datasets. Values of various parameters that have been decided after running the respective steps have been mentioned as and when required. Pictorial description of results using graphs is also included to ensure lucidity for the reader.

4. Chapter-5: **Conclusions and Discussion** mentions in brief a concluding statement about the proposed method. Practical applications of the method have been mentioned. Chapter also briefs about some challenges faced when using our method and mentions about how to possibly rectify them.

Chapter 2

Literature Review

The task of offline signature verification does not have a very strong community of researchers. There exists some researches on this field but they does not provide a uniform standard for performance evaluation.

The task of signature verification is broadly categorized into online and offline signatures, but there exists further distinctions like "exposure to forgery" and "writer dependent/independent".

The exposure to forgery implies what type of signatures were used during the training of the model. Some authors tend to train their models on both genuine and forgery signatures of a person while test their model on the different genuine and forgery signatures of the same person while some train and test on different individuals, some authors do not use forgeries while training their model. The practical application of Signature Verification is very much reflected by the latter methods, as it is very much unlikely to acquire forgeries of an individual.

The signature verification literature includes many different models, many focusing

on feature while some emphasizing on classification method. A few of them have been listed below.

2.1 DRT and HMM

One of the earlier works in the field of verification, Coetzer [2] focus on feature extraction by using the Discrete Radon Transform as a feature extraction technique after some basic image pre-processing methods which are then given as input into a Hidden Markov Model for classification. The system proposes DRT as a feature extraction technique due to its noise reduction and Rotational invariance. Although DRT is not a shift invariant they try to overcast that by image processing methods. Coetzer emphasis on the usage of only genuine signature while training the model due to its practical relevance.

The system trains each model using the Viterbi reestimation technique. When a clam is made that a test pattern belongs to a writer then the pattern is matched with the model using Viterbi algorithm and is quantified by a function. The classification is done by a sliding threshold to determine the error rates. Two independent experiments are conducted on two different data sets with each containing 22 and 51 writers respectively. Experiment used 10 and 15 genuine signatures respectively for training while the remaining genuine along with casual and skilled forgeries for testing. No genuine signatures were used in both the cases for validation purpose. An FRR of $10.2\%, 0.2\%$, an FAR of $25.3\%, 32.6\%$ and an EER of $17.7\%, 4.5\%$ are reported for skilled and casual forgeries on the first data set. While an EER of 12.2% is reported for Amateur forgeries on the second data set.

2.2 DCT-SVM

Shekar et al.[3] Proposes a two stage approach by using Discrete Cosine Transform as a feature extraction method after basic pre-processing of signature images by binarising them using Otsu method and noise filtering by using morphological filters. The features are then fed to a Support Vector Machine classifier. The system proposes 10x10 DCT coefficients as feature vectors. The training and testing samples both include genuine signatures with random and skilled forgery. The experimentation was conducted to publicly available datasets, namely GPDS-160, CEDAR and one regional language database MUKOS. The experiment is conducted on different number of genuine signatures and skilled forgeries in the training sample, with best result reported for 15 genuine and 15 skilled forgery for each writer. An FRR of 7.28% and an FAR of 6.44% is reported on the CEDAR database, while an FRR of 7.63% and FAR of 11.64% is reported on the GPDS-160 database or experimentation with skilled forgeries.

2.3 Machine Learning for Signature Verification

Srinivasan et al. [1] Proposes two learning tasks called person-independent (or general) learning and person-dependent (or special) learning. Feature Extraction and Similarity Comparison approach for both the tasks being same. Methods like Gradient, Structural and Concavity or GSC to get multi resolution features, use of Flexible Templates like landmark mapping and cell alignment by spline mapping is proposed. System very much depends on the baseline features used for training the model to lower the error rates. The model proposes a comparison approach using KL and KS test to detect a forgery in both the tasks, and states a better performance for person-dependent learning as it learns for a specific person. The experiment was conducted on different number of training geniune samples with the best Error Rate 16.07% re-

ported for 17 known genuine signatures. The model also purposes to allow rejection to reduce error rate further to 10%.

2.4 Persian Signature Verification using CNN

One of the early works that proposed the use of deep neural nets in the Signature Verification task. Khalazadeh[4] proposed the use of Convolution Neural Network for the feature extraction method and then using the output or the last layer as an input for the classification task by Multilayer preception network. The network used the LeNet5 as the baseline network. The database used had 176 original Persian Signatures from 22 individuals but lacks skilled forgeries. The model reports an average of 99.86 for validation performance.

2.5 Offline Signature Verification with CNN

Gabe et al.[5] proposes the use of Convolution Neural Net for complete classification task. The model used the transfer learning approach to train the large VGG-16 convolution neural networks, by using the weights trained on the ImageNet database and only allowing the fully connected layers to be trained at the end. The dataset used in the experiment is from the International Conference on Document Analysis and Recognition (ICDAR) 2011 Signature Comp international signature verification competition. The dataset includes both online and offline signatures and is split into training and testing data. Training set used has both genuine and forgery with 25 genuine and 11 forgeries for Dutch dataset and 25 genuine and 30 forgeries for Chinese dataset. The experiment is tested first on ID's which were used while training of the model and reports an FAR of 13.32%, 8.2% and FRR of 3.13%, 18.2% on the Dutch and Chinese dataset respectively. The experiment is also tested on unseen ID's that is individuals that were not in the training set and reports an FAR of 22.2% and FRR

of 21.8% on the Dutch dataset. A writer-dependent comparison is also conducted in which training and testing is done by questioned-reference pairs with each ID having 10 reference signature. This task reports a validation FAR of 33.0% and Validation FRR of 33.0%.

2.6 Writer Independent Feature Learning

Luiz et a [6]. proposes the feature learning approach using Convolution Neural Networks rather than explicit feature extraction procedure. They used a two-stage approach; writer independent feature learning and writer dependent classification. The dataset is divided into two distinct set for this purpose. The Development set is used to train the weights of the Convolution Neural Network for writer-independent feature learning. The resulted function is then used to produce features for the training set which in turn train the writer-dependent classifier. The dataset used for this purpose is GPDS and Brazilian PUCPR dataset. Preprocessing such as background removal using OTSU algorithm, inversion of images, Normalization of the output and resizing the images is also done before giving it as input to the neural network. For training the Writer-Dependent classifiers, no skilled forgeries are used with different number of genuine signatures. The best results are reported for SVM classifier with RBF kernel with an FAR of 5.99% and FRR of 19.81% on the GPDS dataset and an FAR of 13% on Brazilian dataset with CNN trained on the GPDS dataset. Testing is conducted on both skilled and simple forgeries, the output of former being mentioned above. The model achieves low Equal Error rates but has a high False Acceptance Rate and False Rejection Rate.

Chapter 3

Project Work

The project works on verification of a signature from its raw image which is taken as input. From making the image suitable for testing to devising a method to classify it, the process involves several steps, each of which have been described in this section. These steps, namely are:

1. Pre-processing
2. Feature extraction
3. Dissimilarity distribution formation
4. Comparison of dissimilarity distribution
5. Classification

The block diagram of the entire procedure is shown in Figure 3.1.

3.1 Pre-processing

Off-line Signature Verification is done on scanned images depicting signatures. These images generally have the signature on a plain background. Keeping in mind such a

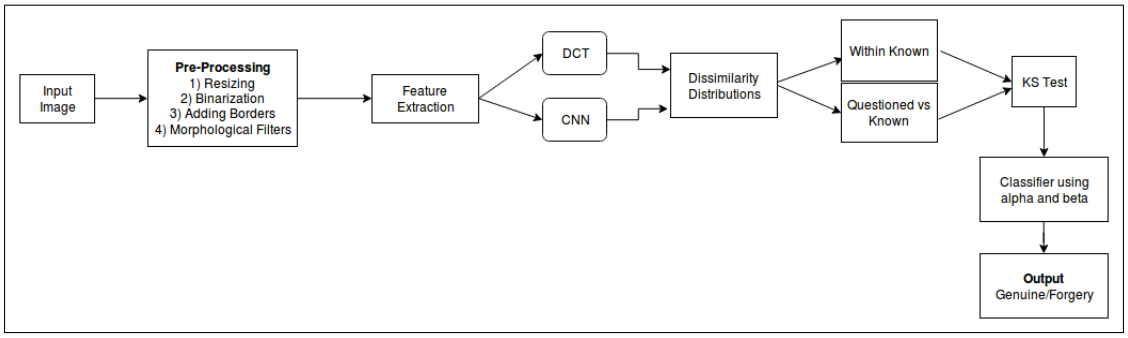

Figure 3.1 Block diagram of the procedure

scenario, we have applied pre-processing techniques for better feature extraction and hence better results. The steps involved in pre-processing in order are:

1. Resizing the raw image with the larger dimension to 100 pixels, keeping the aspect ratio same. See Figure 3.2 for example.

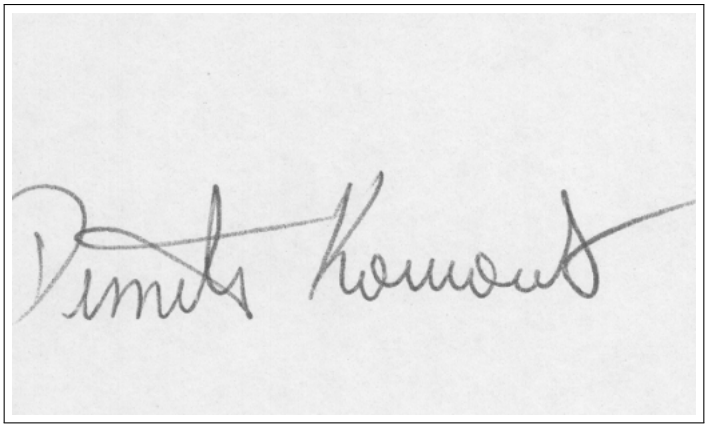

Figure 3.2 Resized grayscale image

2. Application of Otsu binarisation to the resized image in order to transform it to a binary (black and white) image. Refer to Figure 3.3. [3]

3. In order to make the system generic, the image is converted to a square. This is done by adding blank spaces as margins to make the shorter dimension equal to 100 pixels. See Figure 3.4.

4. Application of morphological transformations to reduce noise while keeping the

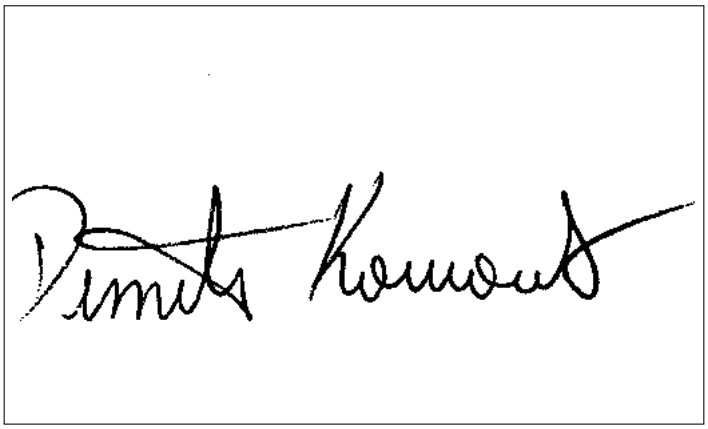

Figure 3.3 Image after binarisation

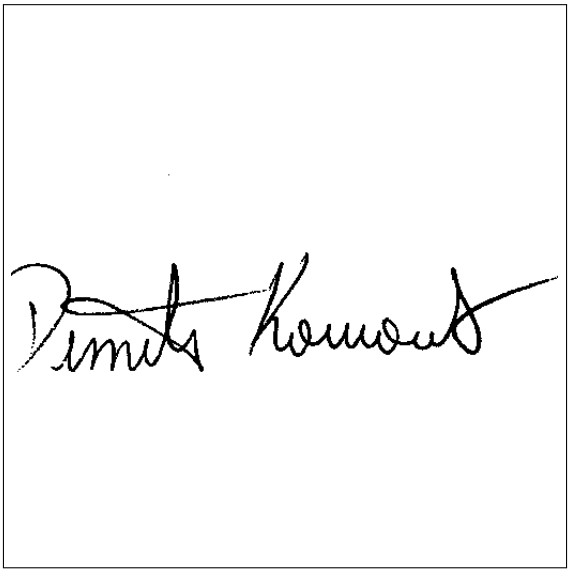

Figure 3.4 Image after adding margins

image area same. We apply erosion to the resultant image in order to unify line width. For instance, see Figure 3.5.

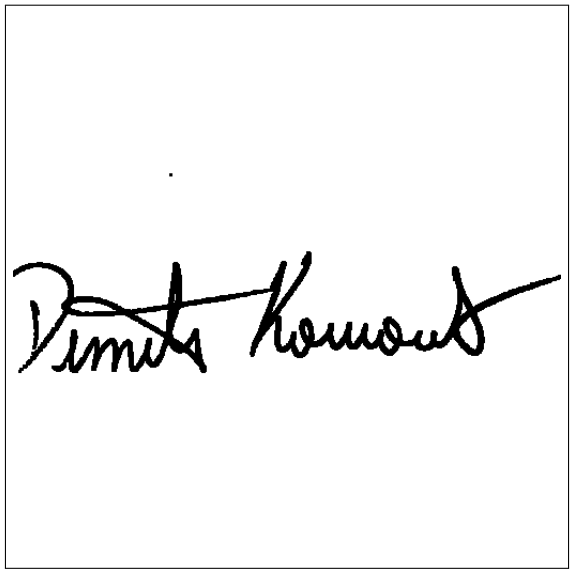

Figure 3.5 Image after Morphological filter has been applied

The entire process of pre-processing has been performed using built-in functions from the OpenCV python library.

3.2 Feature Extraction

Feature extraction in the field of image processing is the process of reducing the amount of resources required to describe a large set of data. Images are represented in the form of pixels. This form of representation of images requires a large amount of memory and very high computation power. Hence, the first and foremost task before applying a machine learning approach is to get the input images in some reduced form called as features. These features should contain relevant information about the image. These features help us get around problems of high computation power and high memory. Features are a representation of the image. Hence better the extracted features, better is the result that we get after learning from those features. Features

should be able to depict details about the image that can be used for classification. We apply and compare two methods for feature extraction:

3.2.1 Discrete Cosine transform

The DCT is a popular technique in image processing and video compression which transforms the input signal present in the spatial domain into a frequency domain.[7] We use 2-D DCT to extract coefficients from the image. The 2-D transform can be represented as:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos \left[\frac{\pi(2x+1)u}{2N} \right] cos \left[\frac{\pi(2y+1)v}{2N} \right] \tag{3.1}$$

The inverse transform is represented as:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u,v) cos \left[\frac{\pi(2x+1)u}{2N} \right] cos \left[\frac{\pi(2y+1)v}{2N} \right] \tag{3.2}$$

It can be noted that the basis functions exhibit a progressive increase in frequency both in the vertical and horizontal direction. Hence, the discriminative features are available in the top-left portion of the image. Due to this, DCT exhibits excellent energy compaction for highly correlated images. It is observed that the DCT coefficients exhibit the expected behavior in which a relatively large amount of information about the original signature image is stored/represented in fairly small number of coefficients. This property comes to aid when extracting coefficients from the transformed image. Features of the image can be extracted using very few coefficients located at upper-left corner. The top $5 \times 10 = 50$ DCT coefficients have been used as a feature vector in our approach.

3.2.2 Convolutional Neural Network

A convolutional neural network (CNN) is a feed-forward network with the ability of extracting topological properties from the input image.[8] Convolutional neural networks combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights, and spatial or temporal sub-sampling. CNN mainly contains following types of layers:

1. A **convolutional layer** is used to extract features from local receptive fields. The convolutional layer is the main layer where all the heavy computational processing takes place.[8] In a regular neural network, neuron of a layer is connected to all the neurons of the next layer. This is impractical and also leads to extremely high requirement of computation levels especially when number of layers increase in the network. Hence, in a CNN, layers have local connectivity i.e. neuron of a layer is sparsely connected, only to a few contiguous section of neurons of the next layer. This is the property of local connectivity in CNN. A convolutional layer is organized in planes of neurons called feature maps. A trainable weight is assigned to each connection, but all units of one feature map share the same weights. In order to extract different types of local features, a convolutional layer is composed of several feature maps.

2. A reduction of the resolution of the feature maps is performed through the **Sub-Sampling layers**. It does not result in change in depth dimension. No new parameters are introduced in this operation. Many different types of sub-sampling layers can be applied including Max-pooling, Min-pooling, average pooling .We have applied sub-sampling by adding a max-pooling layer. Max-pooling benefits us in two ways - It decreases the computational overhead and it also works against over-fitting.

3. **Zero-Padding** is a process of symmetrically adding zeros. It is used when

dimensions of the input layer needs to be preserved in the output layer.[9]

4. **Activation Function** We have used the Rectified Linear Unit (ReLu), the most common activation function for output of the CNN neurons. Mathematically, it is described as:

$$f(x) = max(0, x) \tag{3.3}$$

where x is input to a neuron.[9]

5. **Fully Connected layer** This layer is often added in the last stages. It is used to connect to the output layer and decide the number of outputs.[8]

6. **Softmax layer** This is the final layer that is added in a neural network for classification. Mathematically, it is described as:

$$L_i = -f_{y_i} + \log \sum_j e^{f_j} \tag{3.4}$$

where f_j is the j^{th} element of the vector and L_i is cross-entropy loss.[8]

We note that modeling directly the problem of interest is not feasible in practice as our ultimate goal is to separate genuine signatures from skilled forgeries of the users enrolled in the system, but in a realistic scenario we only have genuine signatures provided during an enrollment phase, and do not have forgeries for these users.

Architecture and training of the network

In order to extract features from a test signature image, we need weights to feed into the neural network. These weights are obtained by training an identical network for the task of classifying a signature into genuine or forged. This network uses both genuine and forged signatures for training. The expectation is that by learning weights for such a classification task, we will be able to apply the same weights to our

classification task that uses only genuines. Our motive is to obtain weights that can express prominent features of the image and that can be achieved by training on our secondary task. This is a case of transfer learning as we are using data learnt form a secondary task in an identical main task.

The secondary task will train weights for the classification of one particular signature. However, the weights must be able to extract suitable features for multiple signatures. Thus, the architecture being used does not contain many layers, unlike typical convolutional neural networks. This ensures the fact that the weights obtained are not very specific to the particular signature.

The network takes as input the 100×100 image. It begins with two pairs of zero padding, convolutional layers. The convolutional layers use the ReLu function as activation. This is followed by a max pooling layer. This triplet of layers is repeated. Next is a fully connected layer that converts the last obtained set of feature maps into a vector of size 700. This vector is then fed into a softmax classifier layer in order to classify the secondary task image as genuine or forged. The parameters of all the layers in sequential order are given below:

1. Zero padding of 1 pixel.
2. 4×4 convolution with 16 filters
3. Zero padding of 1 pixel.
4. 4×4 convolution with 16 filters
5. 2×2 filter max-pooling layer with a stride of 2 pixels
6. Zero padding of 1 pixel.
7. 4×4 convolution with 32 filters

8. Zero padding of 1 pixel.
9. 4×4 convolution with 32 filters
10. 2×2 filter max-pooling layer with a stride of 2 pixels
11. Fully connected layer with 700 outputs
12. Softmax classification layer

The output of the fully connected layer is used as the feature vector for the primary task.

The network is trained with a learning rate of 0.01 for 50 epochs using categorical cross entropy function for the cost. The learning rate has been fixed after trying multiple values and looking for suitable convergence. Other parameters are also obtained after experimenting on different values, descriptions of which have been given in the chapter below on results.

Since the network is not very deep and is to be made not so image specific, we don't require a lot of data for training weights. The data comprises 116 examples (66 genuine and 50 forged) which are split into two parts in a 3:1 ratio for training and testing data.

3.3 Formation of dissimilarity distributions

Once feature vectors are obtained for the training samples and the foreign signature, two distributions of variations or dissimilarities are created. The idea behind doing this is that variations amongst genuine signatures lie in a certain range. This range is captured in these dissimilarity/variation distributions. [1]

All training samples are genuine. However, there exist some variations among these genuine signatures of an individual. These variations are represented by the **within-known distribution** of dissimilarities. Following this, the test sample is compared with each of the training samples. This creates another distribution of dissimilarities known as **questioned vs known distribution**. This distribution accounts for the dissimilarities that the test image holds with the training ones.

These two distributions in distance space are then compared (refer Section 3.4) to check whether the foreign sample is within the range of variation. The test then returns the probability of this sample belonging to the ensemble of knowns.

Consider N training examples to be available for a person. So, the number of pairwise comparisons that can be made are $\binom{N}{2}$. For each comparison, the distance between the two feature vectors is computed. The same process is repeated for all $N_t = \binom{N}{2}$ comparisons. The result is $\boldsymbol{J}$, a $\{\binom{N}{2} \times 1\}$ vector which is a distribution in distance space for the person.

$$\boldsymbol{J} = \{d_1, d_2, \cdots, d_{N_t}\}^\top \tag{3.5}$$

where $\top$ is the transpose operation and d_i is the distance between the samples taken at the i^{th} comparison.

Similarly, a foreign signature is compared with each of the training examples and the distance function is calculated at each comparison to get an $\{N \times 1\}$ questioned vs known distribution vector $\boldsymbol{K}$.

$$\boldsymbol{K} = \{d_1, d_2, \cdots, d_N\}^\top \tag{3.6}$$

Distance function

A distance function is used to obtain the dissimilarity/variation measure. Estimation of such a measure is a major problem in image analysis for which several functions have been proposed. Hausdorff distance, symmetry distance, Euclidean distance , cross correlation ratio are some of them.[10] Here, the euclidean distance has been used as the distance function. The Euclidean distance $d(\boldsymbol{A}, \boldsymbol{B})$ between two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ of size N is given by:

$$d(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{N}\sqrt{\sum_{i=0}^{N}(\boldsymbol{A}_i - \boldsymbol{B}_i)^2} \tag{3.7}$$

3.4 Comparison of dissimilarity distributions

Once we get the dissimilarity distributions of both within person and questioned vs known distributions, our task is to apply a dissimilarity function in order to classify the questioned signature as genuine or forgery. In case the questioned signature is genuine, the questioned vs genuine distribution would be similar to distribution of only genuine signatures. On the other hand, if the questioned signature is a forgery, the distributions of questioned vs. genuines and only genuines would show lesser similarity. This property is used to classify the questioned signature, higher the similarity, higher is the probability of it to be a genuine signature.

For this comparison of dissimilarity distributions, we apply the **Kolmogorov-Smirnov Test (KS)**. The test returns a statistic which quantifies a distance between two unbinned distributions.[1] The distributions should depict functions of a single variable. We have used **KS** test because it has the property of being sensitive about both location and shape of the distributions being compared. The statistic,A, for comparing

two distribution functions $S_{N1}(x)$ and $S_{N1}(x)$ is defined as

$$A = max_{\infty<x<\infty} \mid (S_{N1}(x) - S_{N2}(x) \mid \tag{3.8}$$

The statistic is then mapped to a probability of similarity, P, according to the equation

$$P = T(\sqrt{N_e} + 0.12 + (0.11/\sqrt{N_e})A) \tag{3.9}$$

where T function is given by:

$$T(\lambda) = 2\sum_{j=1}^{\infty}(-1)^{j-1}e^{-2j^2\lambda^2} \tag{3.10}$$

such that: $T(0) = 1$, $T(\infty) = 0$ and N_e is the number of effective data points, $N_e = N_1N_2(N_1 + N_2)$

3.5 Classification

The mentioned KS statistic is used for comparing the two distributions generated to evaluate a probability of similarity between the questioned signature and the reference signatures. Now to make a classification we need to specify a threshold α such that signatures having probability $P > \alpha$ are considered genuine signatures while the questioned signatures having probability $P < \alpha$ are termed as forgeries. The value of α can vary widely on the datasets used as different signatures may require different threshold. To obtain the optimum value of the threshold that is best fit for all the signatures we use ROC curves of False Acceptance Rate vs False Rejection Rate on the validation set.

As proposed by Srinivasan et al [1] we also considered allowing rejections to reduce the Error Rates. The questioned signatures having probabilities lying between the

range $\alpha - \beta$ and $\alpha + \beta$ are rejected and no classification is done on these signatures. As a result this significantly improve the performance as doubtful signatures are not classified leading to lesser error rates. To get the optimum value of α β pair we again use ROC curves.

Chapter 4

Experimentation and Results

The system described above has several parameters which can be tweaked in order to obtain different results. Many of these are altered over various values and the best one has been chosen in order to obtain optimum result. Here, we describe the values for all parameters at various stages and the comparisons done to obtain these values.

Environment

The entire procedure is implemented in Python 2.7 on a Ubuntu 16.04 running machine with a 2 GB NVIDIA GeForce 940M GPU in order to facilitate quick computations, specially those related to deep learning. For deep learning related implementations, Keras library of Python has been used with Tensorflow backend (with GPU support).

Dataset

Datasets used for the evaluation of the model are derived from ICFHR 2010 [11], ICFHR 2012 [12] and CEDAR [13] database.

Dataset A is formed from the ICFHR 2012 [12] with 15 reference genuine signatures for training the model or we can say for comparison with the questioned while has

115 questioned signatures , comprising of genuine signatures and skilled forgeries for that person.

Dataset B is also from the ICFHR 2012 [12] with 16 reference genuine signatures for training the model and 117 questioned signatures.

Dataset C is from the ICFHR 2010 [11] with 25 reference genuine signatures for training and 125 questioned signature different from those used as references.

Dataset D and E are formed by taking two ID's from the CEDAR [13] database with 16 and 24 reference genuine signatures for training respectively and 48 questioned signatures in testing for both the ID's.

The questioned signatures in the Dataset C and D also contains the reference signature used in training.

4.1 Pre-processsing

The input image is resized to 100×100 pixels. The type of binarisation is Otsu algorithm. For the application of morphological transformations, a kernel of size 3×3 is used. This is decided keeping in mind the dimensions of the image.

4.2 Feature Extraction

4.2.1 Discrete Cosine Transform (DCT)

Considering the general shape of signature images, we conclude that the principle DCT coefficients will be found in the top left rectangle. The size of this rectangle is to be decided. A dimension has to be chosen such that neither is the computational cost very high, nor are we compromising a lot on the eventual result. Taking into account these two factors and testing over different number of DCT coefficients, the optimum number was found to be $5 \times 10 = 50$.

4.2.2 Convolutional Neural Network (CNN)

As described in Section 3.2.2, weights for feature extraction in the primary task are trained using a secondary task of binary classification. This task uses dataset B for training weights. As described in Section 3.3.2, the data comprises 116 examples (66 genuine and 50 forged) which are split into two parts in a 3:1 ratio for training and testing data. The batch size used for training is 2.

Various parameters that need to be fixed for this training procedure are:

- Learning rate
- Number of features
- Number of epochs

The learning rate is chosen by varying order of magnitude and looking for suitably quick convergence. Thus, a value of 0.01 is obtained for proper convergence during the classification task training.

In order to decide the number of features and number of epochs, we use the obtained weights for various combinations of these two parameters and test them on the primary task for 3 different datasets. The metric used for testing the performance of weights is the **mean error** which is the arithmetic mean of the **false acceptance rate (FAR)** and the **false rejection rate (FRR)**. For the primary task of classification as genuine or forged these error rates are defined as:

$$\text{FAR} = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{4.1}$$

$$\text{FRR} = \frac{\text{false negatives}}{\text{false negatives} + \text{true positives}} \tag{4.2}$$

The mean error to decide number of epochs and number of features is calculated using $\alpha = 0.05$ and $\beta = 0$. For any value of these variables, the best epochs-features pairs will be the same. The plots obtained for datasets A, B, C for different values of these parameters are shown in Figure 4.1, 4.2 and 4.3 respectively.

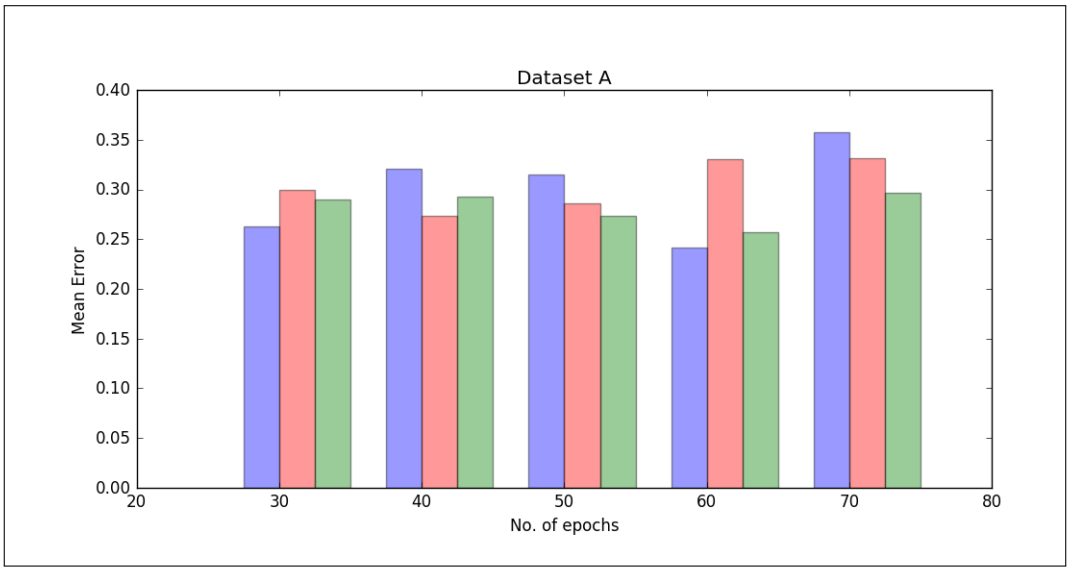

Figure 4.1 Epochs-features plots for dataset A

Looking at the three graphs and nature of individual datsets, it was concluded that 50 epochs, 700 features and 60 epochs, 700 features were the two best choices for training weights to be used in the primary task. During the secondary task of classification, the training accuracy obtained for both of these is 100% and the testing accuracy is 96.55%. The minimum training cost is 1.748×10^{-4} and 1.317×10^{-4}, respectively.

4.3 Dissimilarity Distributions

For the formation and comparison of dissimilarity distributions (described in Sections 3.3 and 3.4), the only variable parameter is the distance function being used. The results of different distance functions vary for different methods of feature extraction. For instance, it is found that the Hamming distance is best suited for the use of binary

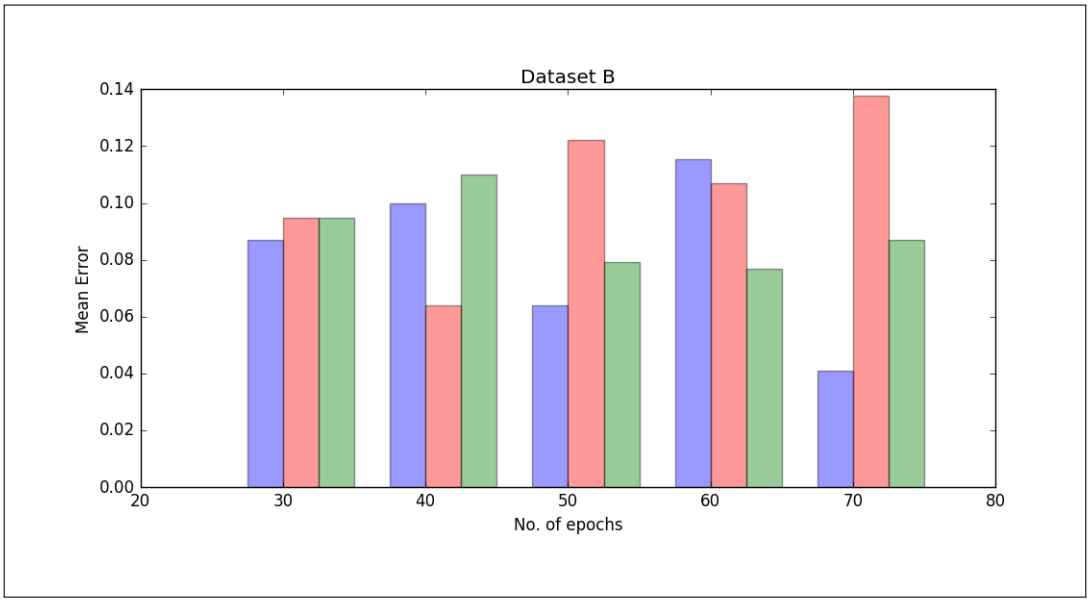

Figure 4.2 Epochs-features plots for dataset B

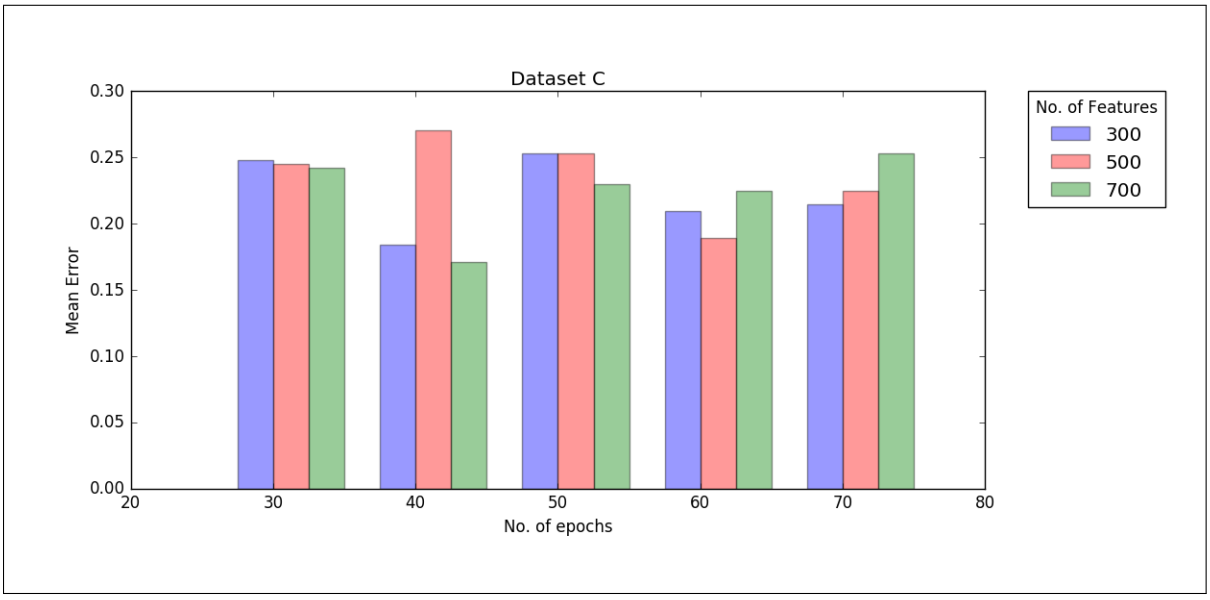

Figure 4.3 Epochs-features plots for dataset C

GSC features [1]. Here, we use DCT and CNN-extracted features. Several distance functions were tried on these features as listed in Section 3.3. The performance of each of these was evaluated using the metrics given in equations 4.1 and 4.2. It was found that the Euclidean Distance was the most suitable to be used under the given circumstances.

4.4 Classification

After finalising the parameters for feature extraction and the distance function, what remains is to fix a value for α and β in order to perform the final classification based on the probability obtained in the KS test. To obtain an optimum value of α, we use different values in an interval and evaluate the performance using FAR and FRR metrics. This is done for datasets A, B and C without the use of β for DCT features (Figure 4.4) and CNN-extracted features (Figure 4.5). It can be observed from the plots that the optimum value of α is 0.03 for DCT features and 0.11 for CNN-extracted features in the case of 50 epochs, 700 features.

Trade-off between FAR and FRR

For the evaluation of performance on different values of α, we don't use mean error as a metric because in many cases, the FAR and FRR vary widely. The use of mean error in such situations does not provide the complete picture. It is important that in order to minimize the overall mean error, we don't compromise with one of the error rates a lot.

Coming to which one out of FAR and FRR is a more dominant factor in taking a decision, it seems logical that for the problem of signature verification it is not a serious problem to classify a genuine signature, unlike classifying a forgery as a genuine sample. The latter one is dangerous for the security of the area of application.[5]

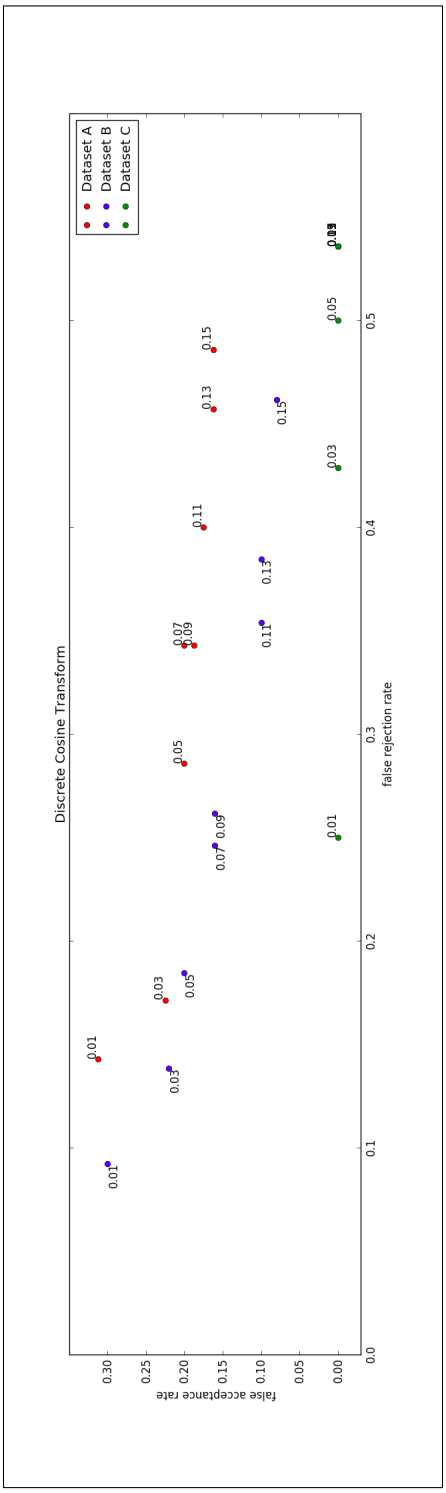

Figure 4.4 ROC plot for DCT features

Figure 4.5 ROC plot for CNN-extracted features

Hence, we take care that neither one of FAR and FRR is incoherently high. However, while taking decisions where these rates are at a comparable scale, the parameter giving a better value of FAR is chosen.

The better method of feature extraction

Having obtained the best values of α for DCT and CNN extracted features, the results obtained for both of these on datasets A, B, C can be tabulated as in Table 4.1.

Evaluation Metric	DCT ($\alpha = 0.03$)			CNN ($\alpha = 0.11$)		
	A	B	C	A	B	C
FRR	0.172	0.139	0.429	0.285	0.277	0.464
FAR	0.225	0.22	0	0.162	0	0.020
Mean error	0.199	0.180	0.215	0.223	0.138	0.242

Table 4.1 Comparison of results for DCT and CNN-extracted features

The results for DCT and CNN extracted features are close as far as the mean error is concerned. One might also conclude that DCT is better. However, there is a stark difference in FAR which is a lot better for CNN extracted features. As far as the FRR is concerned, the comparatively worse values for CNN are not so high that they outweigh the disadvantage that DCT caries due to high false acceptance rates. Hence the final parameter decided were $\alpha = 0.11$ for 700 CNN extracted features with weights trained for 50 epochs.

4.5 Final Results

Since, now we have the final values of all parameters, we increase the performance of the system by using the factor $\beta = 0.1$ and testing on more datasets. After using this parameter, results improve and after testing on 2 more datasets, the results for different values of α are plotted in Figure 4.6. As expected the beast value of α

remains the same. The final results are hence tabulated in Table 4.2 for 700 CNN-extracted features using weights trained for 50 epochs. α used is 0.11.

Evaluation Metric	A	B	C	D	E
FRR	0.1	0.025	0.38	0.2	0.2
FAR	0.15	0	0.021	0.11	0
Mean error	0.125	0.013	0.201	0.155	0.1

Table 4.2 Final performance results after incorporating β

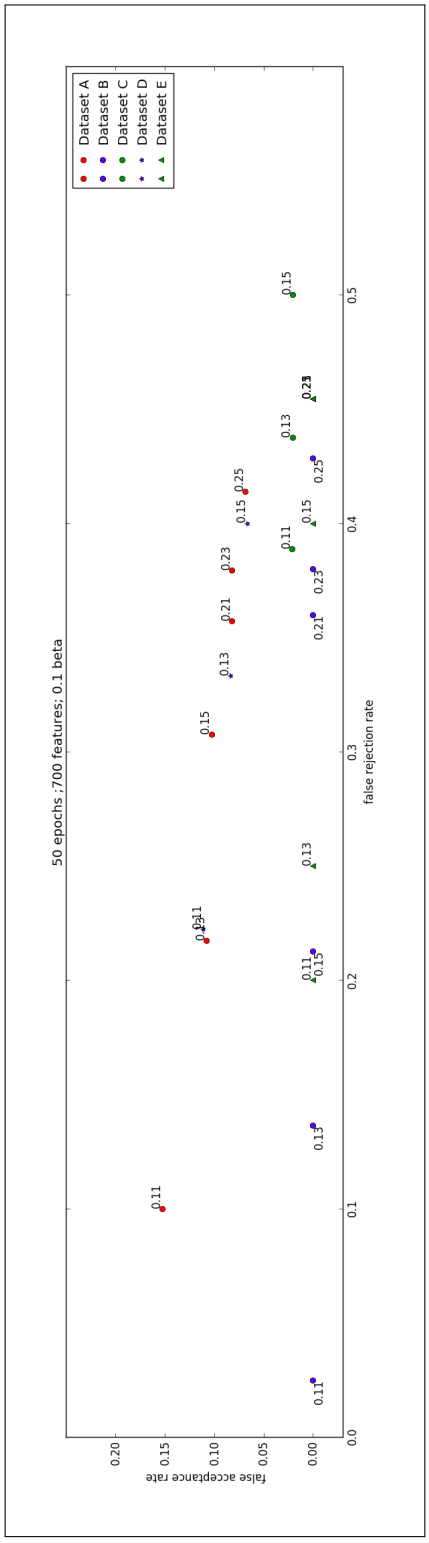

Figure 4.6 Final results after incorporating β

Chapter 5

Conclusions and Discussion

We presented a framework for the Offline Signature Verification task which does not require the usage of defining explicit feature extraction techniques that widely vary on the dataset being used. The framework also does not require to use forgeries while classifying a questioned signature which very well reflect the practical application of the task. Experiments conducted on the CEDAR dataset using this model produces result better than previous work.

Future Directions

We note that on training the Convolution Neural Network the weight trained somewhat specifies to that signature, this can be improved if we use Auto-encoders instead of classifier to train the Neural Network, since then it would produce more general results.

Lastly the use of beta to produce a region of uncertainty in prediction and rejecting the image helps in improving the Error Rates and very well coincide with the practical application. This can be further explored to produce optimum results.

Bibliography

[1] H. Srinivasan, S. N. Srihari, and M. J. Beal, "Machine Learning for Signature Verification," in *Machine Learning in Document Analysis and Recognition*, 2008, pp. 387–408.

[2] J. Coetzer, B. M. Herbst, and J. A. du Preez, "Offline Signature Verification Using the Discrete Radon Transform and a Hidden Markov Model," in *EURASIP Journal on Applied Signal Processing*, 2004, pp. 559–571.

[3] B. H. Shekar and R. K. Bharathi, "DCT-SVM-Based Technique for the Off-line Signature Verification," in *Emerging Research in Electronics, Computer Science and Technology*, 2013, pp. 843–853.

[4] H. Khalajzadeh, M. Mansouri, and M. Teshnehlab, "Persian Signature Verification using Convolution Neural Network," in *International Journal of Engineering Research and Technology*, 2012, p. Vol 1 Issue 2.

[5] G. Alvarez, B. Sheffer, and M. Bryant, "Offline Signature Verification with Convolution Neural Network," 2016.

[6] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Writer-Independent Feature Learning for Offline Signature Verification using Deep Convolutional Neural Networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2016.

[7] S. A. Khayam, "The Discrete Cosine Transform (DCT): Theory and Application". Department of electrical computing engineering," 2003.

[8] "Convolutional neural networks for visual recognition," in *http://cs231n.github.io/convolutional-networks/*.

[9] "Convolutional neural networks (cnns): An illustrated explanation," in *http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/*.

[10] V. Di Gesu and V. Starovoitov, "Distance-Based Functions for Image Comparison," in *Pattern Recognition Letters 20*.

[11] M. Liwicki, E. Heuvel, B. Found, and M. I. Malik, "Forensic Signature Verification Competition 4nsigcomp2010 – Detection of Simulated and Disguised Signatures," in *Proc. 12th Int. Conference on Frontiers in Handwriting Recognition*, 2010.

[12] M. Liwicki, M. I. Malik, E. Alewijnse, L Heuvel, and B. Found, "ICFHR2012 Competition on Automatic Forensic Signature Verification (4Nsigcomp 2012)," in *Proc. 13th Int. Conference on Frontiers in Handwriting Recognition*, 2012.

[13] M. K. Kalera, S. N. Srihari, and A. XU, "Offline Signature Verification and identification using Distance Statistics," in *International Journal of Pattern Recognition and Artificial Intelligence*, 2004, pp. 1339–1360.