OFFLINE SIGNATURE RECOGNITION USING DEEP CONVOLUTION NEURAL NETWORK

Thesis submitted by

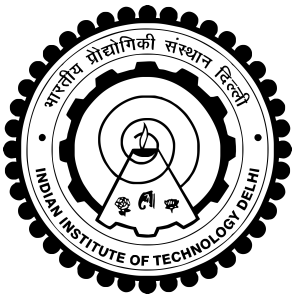
Priyank Soni
2016EET2639

under the guidance of

Prof. Dr. A.P. Prathosh

in partial fulfilment of the requirements
for the award of the degree of

Master of Technology



Department Of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

May 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **OFFLINE SIGNATURE RECOGNITION USING DEEP CONVOLUTION NEURAL NETWORK**, submitted by **Priyank Soni**, is a bonafide record of the research work done at the Indian Institute of Technology, Delhi, for the award of the degree of **Master of Technology**, under our supervision. The contents of this thesis, either in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Prof.
Dr. A.P. Prathosh
Dept. of Electrical Engineering
IIT-Delhi, 110016

Place: New Delhi

Date: 20th May 2018

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Dr.A.P. Prathosh for his support and guidance. He has been my source of motivation for doing this research work.I would also like to thank Electrical Engineering Department of IIT Delhi for providing me opportunity of getting invaluable education. I feel deep respect for my parents for providing constant support and motivation throughout my life, I would be nothing without their help and guidance.Finally thanks to my friends and colleagues for motivating me time to time. I bow to almighty for giving me enough strength for doing this work.

ABSTRACT

Offline handwritten signature verification is an active research area for last few decades in computer vision, signal processing, graphology communities. Handwritten signatures are one of most important bio-metric traits which can be used to uniquely identify a person, though there may be intra class variability but they can be verified by some unique features in curls, free ends etc. In spite of the research advancements it's still very hard to classify a genuine signature and skilled forgery (targeting genuine signature).

People tried to extract different feature from signature (height, width, best fit, DCT, wavelet transform, etc), here we are trying to learn signature features using Deep Convolution Neural Networks.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Motivation

Signature verification research is mostly divided into: online (dynamic) signature verification and offline (static) signature verification.

In online method, signature is captured using an electronic device like tablet, etc. Where information like speed, pressure, pen inclination, coordinates ,etc. are recorded and can be used as features, but in offline method signature is taken on a piece of a paper that is scanned and converted to a digital image, now we need to analyze that image. Handwritten signature recognition is very interesting problem as method of collecting is non invasive and as everyone is familiar with this and uses this in variety of places (legal, administrative and financial documents).

1.2 Objective

The primary objective here is to design a system for identification and verification of user based on his offline handwritten signature. Signature recognition systems may be writer independent(WI) system or writer dependent(WD) system, In WI systems there is only one system for all users and in WD system there are as many systems as the number of users enrolled. In combined approach final decision is given by combining the results of the WI and WD systems. Generally WI system is a kind of feature learner which transforms the given input (signature) into another feature space, now these feature vectors can be used for WD systems and at the time of operation final decision can be given only using the WD systems. Here we design a WI system for getting signature features only. Fig 1.1 shows the overall system configuration used for offline handwritten signature verification purpose

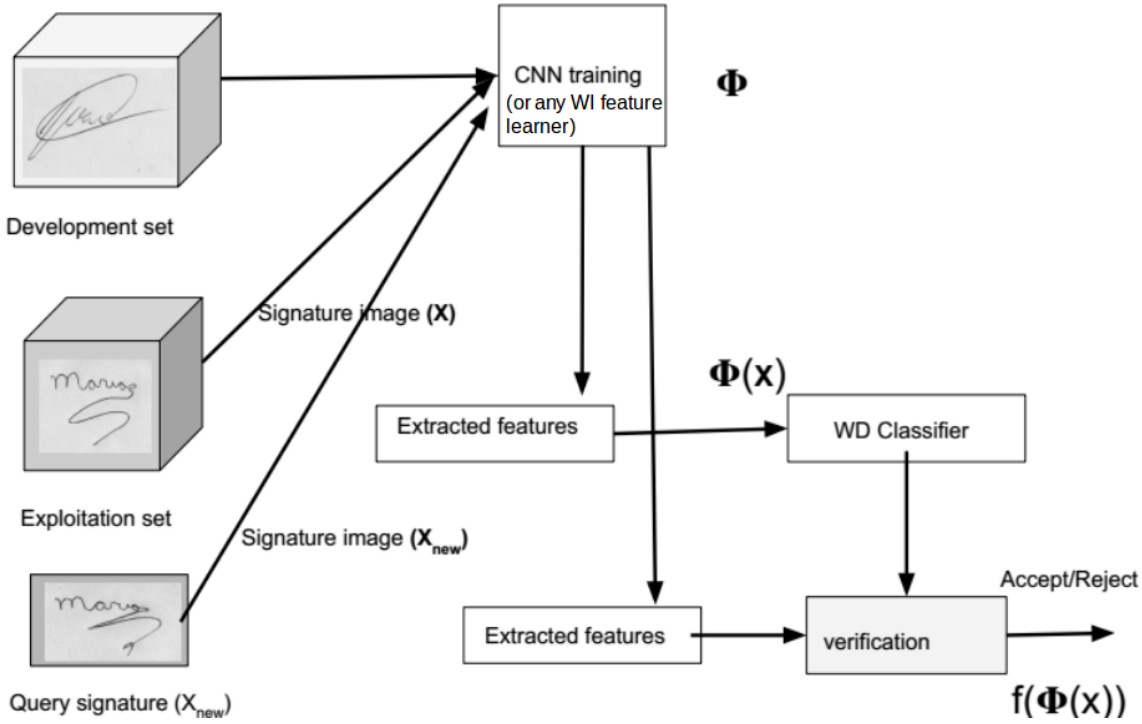


Figure 1.1: Signature verification using combined WD and WI systems

1.3 Organization of thesis

The complete thesis is organized in five chapters. Chapter 1 is overview of our problem, chapter 2 explains some recent work in area of offline handwritten signature verification, chapter 3 contains all preliminaries required for our task, chapter 4 is about implementation details, chapter 5 shows experiments, results, conclusion and future scope of research in offline handwritten signature verification.

Chapter 2

LITERATURE REVIEW

The bio metric system is used to uniquely identify the person based on physiological traits like face, iris, fingerprint or behavioural trait like voice, handwritten signature. In verification problem sample is taken from user and system tells that he is actually the person who claimed to be, while in identification, system identifies the user among all the users [15].

The task of signature verification system is to classify a given sample as genuine of forgery. There are 3 kind of forgeries: (1) Random : where user is unknown person to the forger means he has no idea about his signature. (2) Simple: where forger knows user's name but don't know his signature. (3) skilled: where forger knows both name and signature of the user, In this case it's tough to classify this as a forgery.

2.1 AIM

Here the task to be done is verification of the signature, where we have few samples available from the users and we need to design a system which can classify the query signature as genuine or forgery. On the basis of quantity of the available data we can design a WI system or WD system, for designing WD system we need more data per person so that a single system can be designed for each individual. In case of WI system we take subset from complete dataset (development set D) which we use to train WI classifier and exploitation set E, which represents registered users in the system can be used to design WD system.

2.2 Signature Datasets

There are several publicly available signature datasets. For collecting the data the user is provided a form with boxes of appropriate sizes, in which he does his signature after that the form is scanned (at 300 or 600 dpi). Some datasets are following:

Datasets we are using for our research are GPDS960 and GPDSSyntheticSignature database.

Table 2.1: Some publicly available signature databases

Name	*samples*	*genuine signatures*	*forgeries*
MCYT-75	75	15	15
CEDAR	55	24	24
GPDS960	881	24	30
GPDS Synthetic signature	10000	24	30

2.3 Difficulties faced in signature verification task

The main challenge in signature verification system is high intra class variability in signatures. Fig.2.1 shows an example for illustration.

Figure 2.1: Signatures of same person superimposed showing high intra class variability

In skilled forgery case, its difficult to classify because of low inter class variability. One more main problem is that in practical scenarios during training we have only genuine signatures present but our system needs to reject forgeries from genuines. Also we have only few samples available of each user.

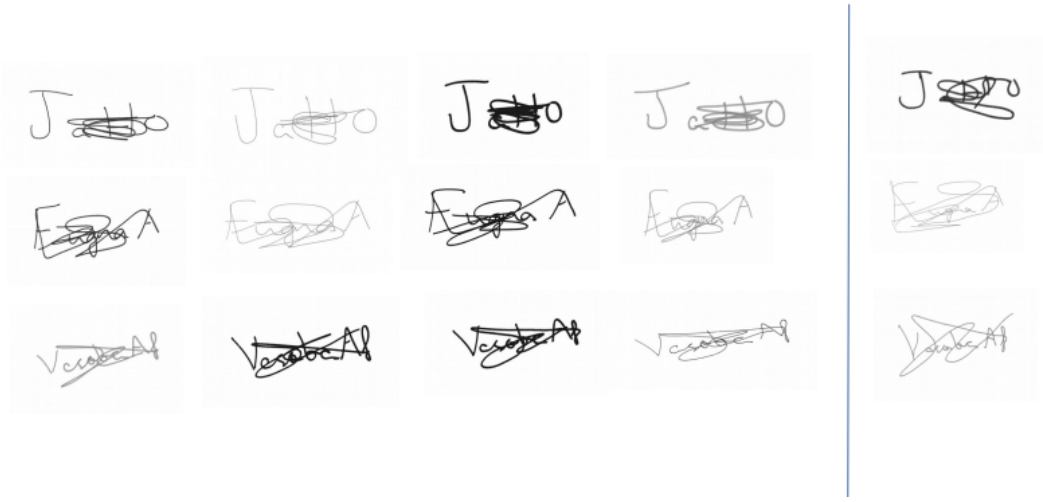


Figure 2.2: Every row at left side shows Signatures of same person with some difference (showing high intra class variability), and at right side skilled forgery (showing very low inter class variability)

2.4 Handcrafted Features

Signature feature can be divided broadly into global and local feature , where global features represents signature's height, width, width to height or height to width ratio, projections, area, centroid, kurtosis, variance, mean, std. deviation, eccentricity, etc. ,where as the local features are pattern at a particular grid position, thickness of the line, curls, angels of free end of the signature, etc. Others categories are grid features, mask features and texture features.

Khalifa [3] used grid, global and texture features and trained 3 NN in parallel and a combiner. Bajaj [4] used 3 MLPs for moment features, upper, lower envelop features.

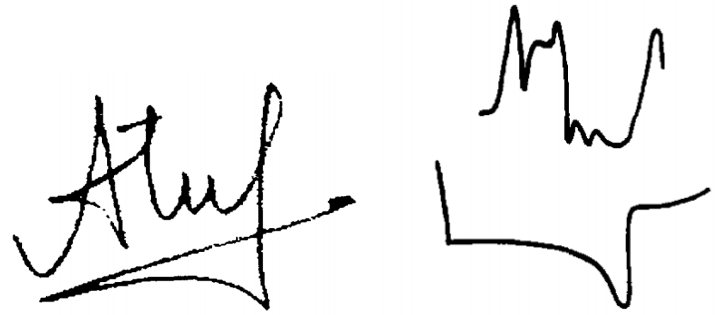


Figure 2.3: Sample signature-1 and it's upper and lower envelope

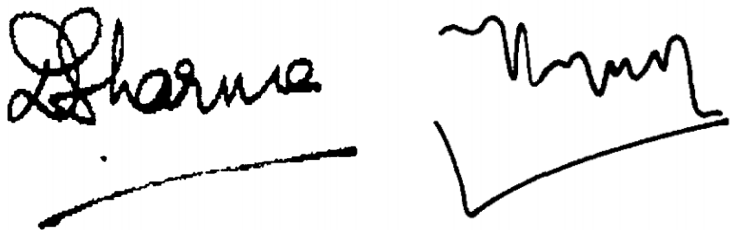


Figure 2.4: Sample signature-2 and it's upper and lower envelope

Karanjkar, Vasambekar [5] worked on signature recognition on bank cheques by extracting ROI by cropping and considered features like variance, mean, std. deviations, eccentricity, kurtosis and trained NN with back propogation. Chauhan [6] considered std. deviation, orientation, area, entropy and eccentricity to make 5 dimensional feature vector and trained neural network.

2.4.1 Using deep learning

Here instead of relying several on hand crafted features, features are learned with help of the raw data [7], deep architecture are well suited for learning abstractions in vision tasks.

Alvarez [8] worked on Dutch and Chinese signature where they considered pairs of two genuine and 1-genuine, 1-forgery for training. Ribeiro [9] worked with multiple classifiers and used k-means, DCT, DWT for getting features.

In recent approach a subset of all users is used for learning important signature features [10], a feature extractor (WI) is designed and using feature vectors extracted from this system we train individual classifiers (WD) [14] for each user which is a binary classifier. Generative adversarial Nets proposed by Goodfellow [11] are also being used for this task. DCGAN [12] proposed by Radford and Chintala are good candidates for learning image features, here we have two networks generator and discriminator, images generated by generator are assessed by discriminator, finally generator generates good images. Feature can also be extracted using intermediate layers of theses networks [13].

2.4.2 Performance on some commonly used datasets

For performance comparison we need some standard datasets and some similar experimental protocols, but training protocols may vary like how many references we are using for training and selection of random forgeries for training and also testing protocols like matrices (FAR, FRR, EER, etc) to be reported.

Table 2.2: Some results on GPDS databases

System	Features,algo	No.of References	$FAR_{skilled}$	FRR	AER	Dataset
WD[1]	Wavelet, SVM	-	5.9	24.8	15.3	GPDS750
WD[2]	MLP, SVM	-	8.94	8.59	-	GPDS300
WD[10]	Feature learning, SVM	12	3.53	3.94	3.74	GPDS300
WD [8]	DCGAN	14	-	-	16.8	GPDS960

Chapter 3

Basic preliminaries

3.1 Convolutional Neural Network

Like neural networks, Convolutional Neural Networks also are made up of neurons, also have weights, biases. Input for each neuron is the weighted sum, which is passed through activation function for getting an output.The network has a loss function and all tricks used for neural networks works here also [16].

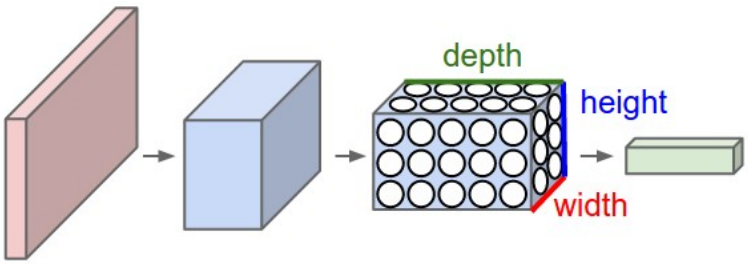


Figure 3.1: Neurons arranged in 3 dimensions. A 3D input volume(here red input image) is transformed into 3D output volume

3.1.1 Layers in Convolutional Neural Networks

For building convNets mainly 3 types of layers are used: convolutional, pooling , and fully connected layers (same as used in neural nets). Theses 3 layers are stacked for building a convNet architecture.

Convolutional layer

Core building block of CNN is convolutional layer. It consist kernels (filters), where each filter convolved across the height and width of the input volume, dot product between input and filter is computed in forward pass and 2D activation map is produced, finally network learns the filter values which gets activated when exposed to certain kind of features in input.Full output volume is formed by stacking activation maps for all filters along depth [17].

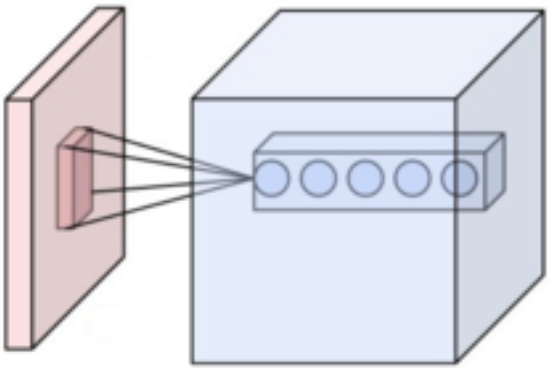


Figure 3.2: Neurons in convolution layer connected to the receptive field

Pooling layer

Here pooling can be considered as down sampling in non-linear form. Several ways of pooling are there but max pooling is widely used. Input image is partitioned into non overlapping rectangles and maximum is taken for each section. The pooling layer resizes the input by operating on the each depth slice and results in resizing the input. The intuition is that absolute location of the features is not important but its important relative to the other features, the pooling layers helps in reducing parameters and computations and so controlling the overfitting [32].

Other than max pooling there are some variations also used like average pooling, L2-norm pooling, another variant is ROI (region interest pooling) where output size is always fixed.

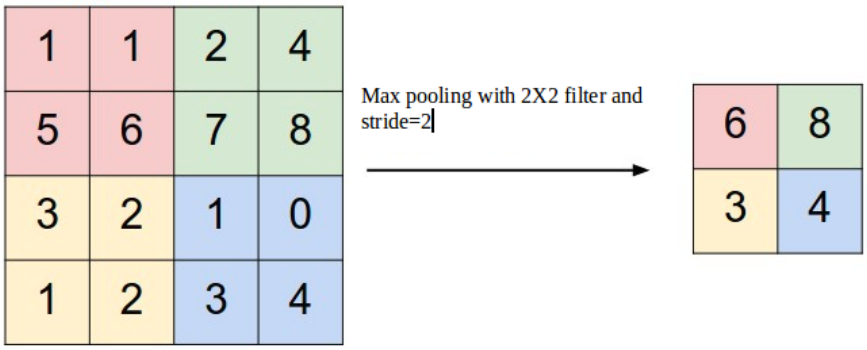


Figure 3.3: Max pooling with 2X2 filter and stride=2

Fully connected layer

Neurons in a fully connected layer has connections to activations in previous layer, which acts as input to the softmax layer which has nodes equal to the number of classes present

and gives output between 0 and 1 as probability.

3.2 VggNet

It was developed by Zisserman and Simonya, it was runner up in ILSVRC 2014 competition. It consist 3X3 convolution layers and too many filters. Weights of vggNet are available publicly which can be used for transfer learning in our models.

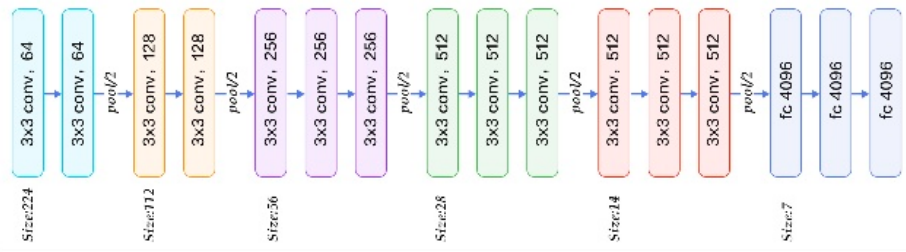


Figure 3.4: vgg16 model architecture

Table 3.1: VggNet model dimensions

Layer name	No.of filters	stride	output shape
Convolution	64	1	224X224X64
Convolution	64	1	224X224X64
Maxpool	-	2	112X112X64
Convolution	64	1	112X112X128
Convolution	64	1	112X112X128
Maxpool	-	2	56X56X128
Convolution	64	1	56X56X256
Convolution	64	1	56X56X256
Convolution	64	1	56X56X256
Maxpool	-	2	28X28X256
Convolution	64	1	28X28X512
Convolution	64	1	28X28X512
Convolution	64	1	28X28X512
Maxpool	-	2	14X14X512
Convolution	64	1	14X14X512
Convolution	64	1	14X14X512
Convolution	64	1	14X14X512
Maxpool	-	2	7X7X512
fc	4096	-	1x1x4096
fc	4096	-	1x1x4096
fc	4096	-	1x1x4096

3.3 ResNet

ResNet model was developed by Kaiming, as we know that when overfitting is handled ,then increasing the number of layers will increase the accuracy, but there is the problem of vanishing gradients. ResNet has 'skip connections'. Here increasing the number of layers does not create problem because of the identity mapping, the performance of the model will not be less than the shallow network. There are several variants available of residual block also.

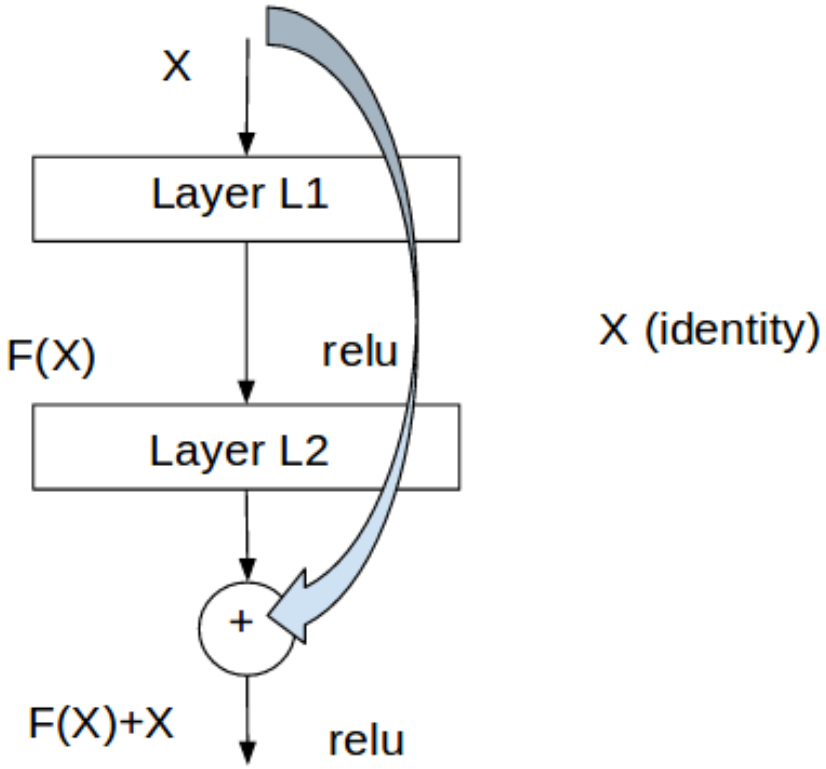


Figure 3.5: A residual block

3.4 Support Vector Machine

SVM is used for getting optimal hyperplane for the patterns which are linearly separable and also for non-linear after suitable transformation (kernel function), data points closest to the hyperplane are called support vectors, they are very critical points so removing them may change the hyperplane also. Hyperplane is specified by the locations of the support vectors as shown in the figure below[33].

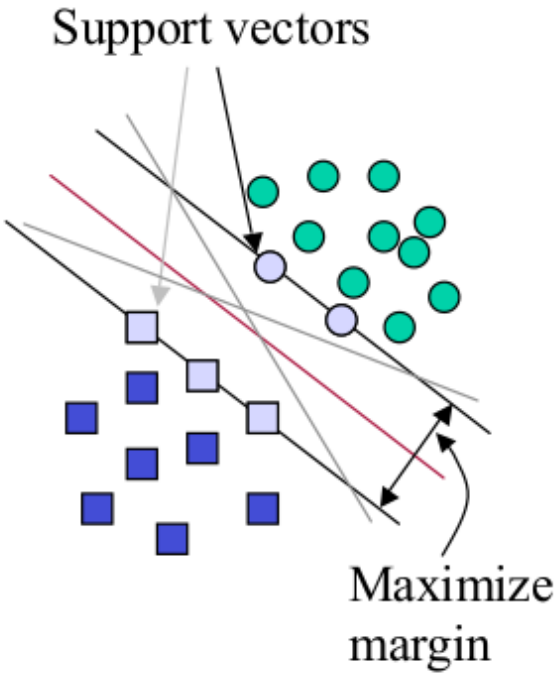


Figure 3.6: SVM hyperplane

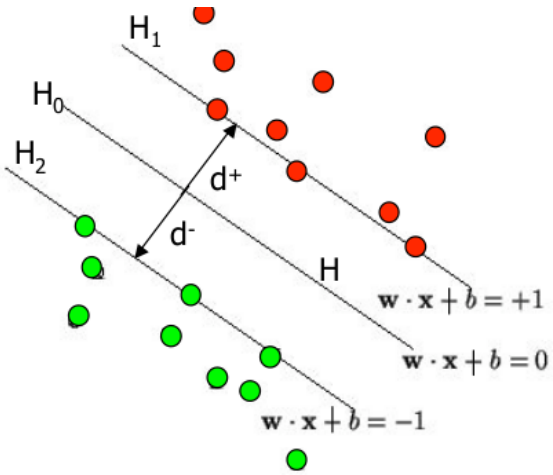


Figure 3.7: Hyperplane equations

In fig-3.8 ,the hyperplanes H can be defined as:

$$W.X + b >= +1 \quad \text{if } Y_i=+1$$

$$W.X + b <= +1 \quad \text{if } Y_i=-1$$

Here for H1:

$$W.X + b = +1$$

and for H2:

$$W.X + b = -1$$

3.5 Adaboost

Here results from weak classifiers are combined for giving the final result. For weak classifier the error rate is little bit better than random. In each iteration we train M classifiers, the samples which are getting miss classified will get higher weights next time [30].

1. Initialization W_i=1/N, i=1,2,.....N

2. for m=1 to M:

(a) Fit classifier $G_m(x)$ to training data with weights W_i

(b) calculate $err_m = \frac{\sum_{i=1}^{N} W_i I(y_i \neq G_m(x))}{\sum_{i=1}^{N} W_i}$

(c) Compute $\alpha_m = log((1 - err_m)/err_m)$

(d) $W_i = W_i \exp[\alpha_m I(y_i \neq G_m(x))]$

3. Output $G_m(x)$ =sign$[\sum_{i=1}^{N} \alpha_m G_m(x))]$

Chapter 4

Implementation Details

Here the idea is to divide whole dataset into two parts: Development set (D) and Exploitation set (E), where E represents enrolled users in the system. D is used to learn representation from signature by training a CNN (WI classification) or DCGAN. Once CNN is trained, this CNN is used as feature extractor and writer dependent binary classifier is trained for each user (WD classification).

4.1 Pre-processing

The size of signature images given in GPDS960 and GPDSsynthetic signature databases are varies from 154x268 pixels to 817x1141 pixels. We need all images to be of same sizes to train a neural network, here we applied same prepossessing steps to images in Development and Exploitation sets (D and E)

4.1.1 Adding image to the canvas and noise removal

Here we are taking a canvas of size (840X1360), Now we find centre of mass of the image and put this image on centre of the canvas, in this process we need to make sure that image is not going out of the bounds, if this happens we need to crop the height or width whichever is extra. Now put image on the canvas.

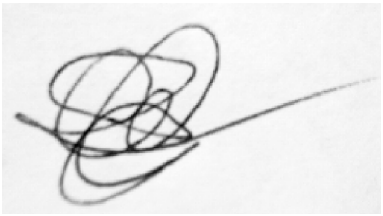


Figure 4.1: Original image.

OTSU's algorithm was used for finding the threshold between the background and foreground intensities.If pixels intensity is greater than threshold then set to intensity 255 (white).

4.1.2 Inverting and resizing

We are inverting the image as:

$$Image_{\text{inverted}} := 255 - Image$$

so that our background becomes black. Now we resize the image to 170X242 using linear interpolation without stretching the image.

Figure 4.2: Image after binarization

4.1.3 Cropping

Finally we crop all images to size 150x220, for giving them as input to the neural network

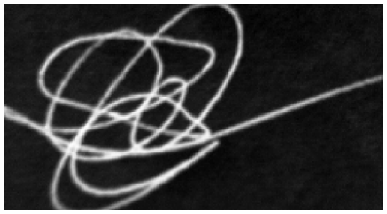


Figure 4.3: Cropped image

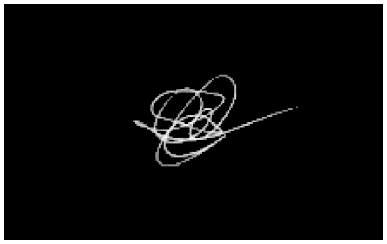


Figure 4.4: Final image.

4.2 Writer Independent representation learning with CNN

If we have large number of samples available per user then training classifiers for each user is a good idea, but generally we only few (20-25) signature samples available per user, so only WD systems will not lead good results. In GPDS960 and GPDSSyntheticSignature databases where we have data from total 10000 user, it's beneficial to design a WI feature learner.We split our dataset into Development set (D) and Exploitation set (E). Now since CNNs are better candidates for image data, we train a CNN (WI) for learning useful features of the signature (curl, free ends, etc).The output of the fully connected layer before final layer is $\boldsymbol{\phi(X)}$ which is representation space and signatures will be linearly separable in this space. Now samples are transformed into another feature space($\mathbf{X}$ to $\boldsymbol{\phi(X)}$) and these feature vectors can be used to train WD classifiers (SVM, Adaboost).

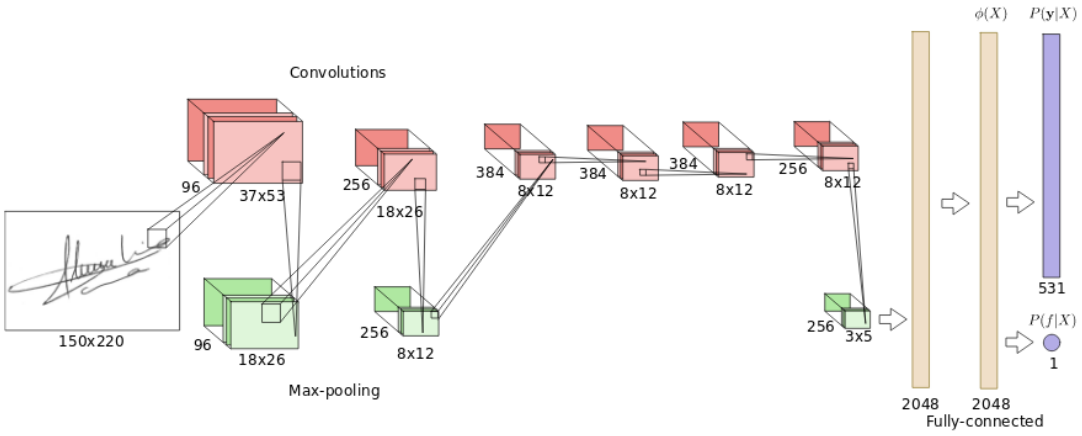


Figure 4.5: CNN used as a feature extractor for images in Exploitation dataset

4.2.1 Using only genuine signatures

Let's say there are $\mathbf{Y}$ users in development set, we train CNN with genuine samples from all $\mathbf{Y}$ users.Every training example is a tuple (X,y), where y is user y ϵ $\mathbf{Y}$ and $\mathbf{X}$ is an image. Here we have total of $\mathbf{Y}$ user, so our last layer will contain $|\mathbf{Y}|$ softmax unit, each node gives probability of sample being in that class.

Our neural network is trained to minimize the loss function:

$$\mathbf{L}=-\sum_{j=1}^{m} y_{\mathrm{i,j}} log P(y_j/X_i)$$

Where $P(y_j/X_i)$ is probability that signature X_i belongs to the class y_j. For given example X_i true class is $\mathrm{y_{i,j}}$.

4.3 Writer Independent representation learning with DCGAN

Here we train Deep Convolutional Generative Adversarial Network (DCGAN) [12] using data in development set. DCGAN's are good candidates for extracting representation of images.There are two models we need to train, first is discriminator model which is trained with both true samples and samples generated by generator (false), second is adversarial model which is combination of generator followed by discriminator, this model is trained with the samples generated by the generator with giving them true lable [18] . After training, we use intermediate layers from discriminator for getting representation for our signatures. Now WD systems are trained using extracted features which may be SVM or Adaboost classifiers.

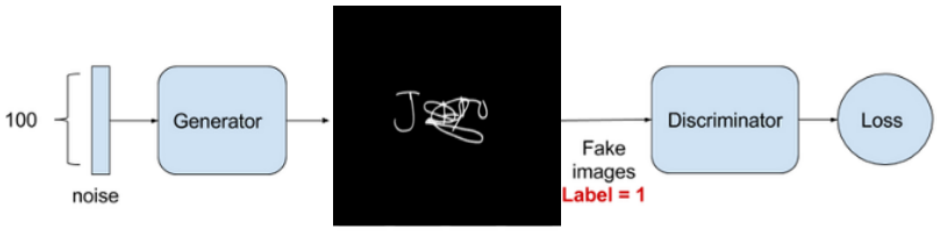


Figure 4.6: Adversarial model training

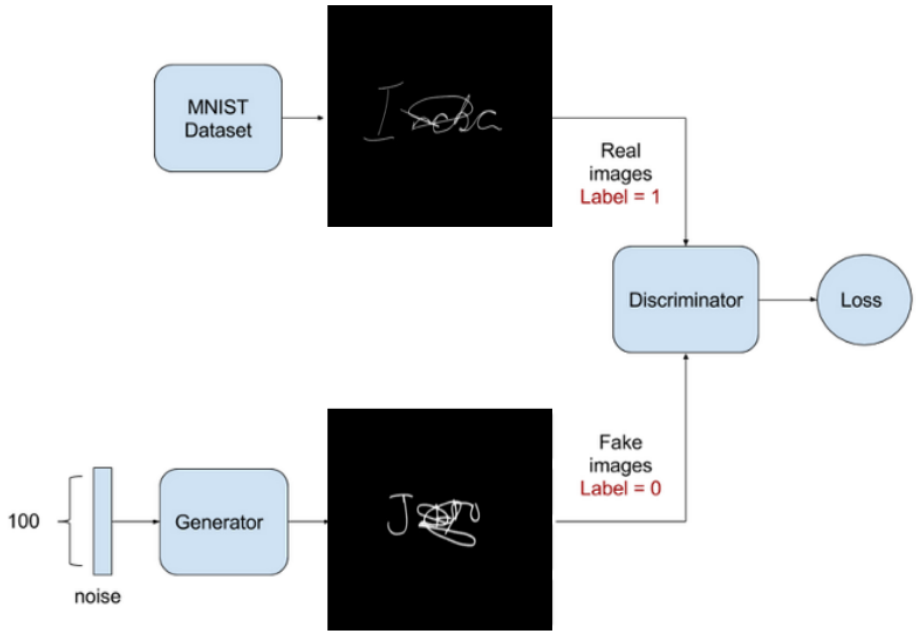


Figure 4.7: Discriminator model training

4.4 Writer Dependent Classifier

Here we train a binary classifier for each user using SVM, Adaboost classifier. For training a WD classifier signature images ($\mathbf{X}$) are fed to WI system and corresponding feature vectors ($\boldsymbol{\phi}(\mathbf{X})$) are obtained, which are now used for training.

Chapter 5

Experiments and Results

Here we are including the details of all the experiments done along with their results. Datasets we are using for training our models are GPDS960 which contains samples from 881 user and GPDSSyntheticSignature databases which contains samples from 10000 user both databases have 24 genuine signatures and 30 forgeries for each user. GPDS dataset contains samples from 881 users, 1-300 considered as Exploitation set and rest as Development set,In GPDSSyntheticSignature database we are experimenting with 1000 samples, a CNN is trained with Development dataset with the following architecture for projecting signature image into feature space:

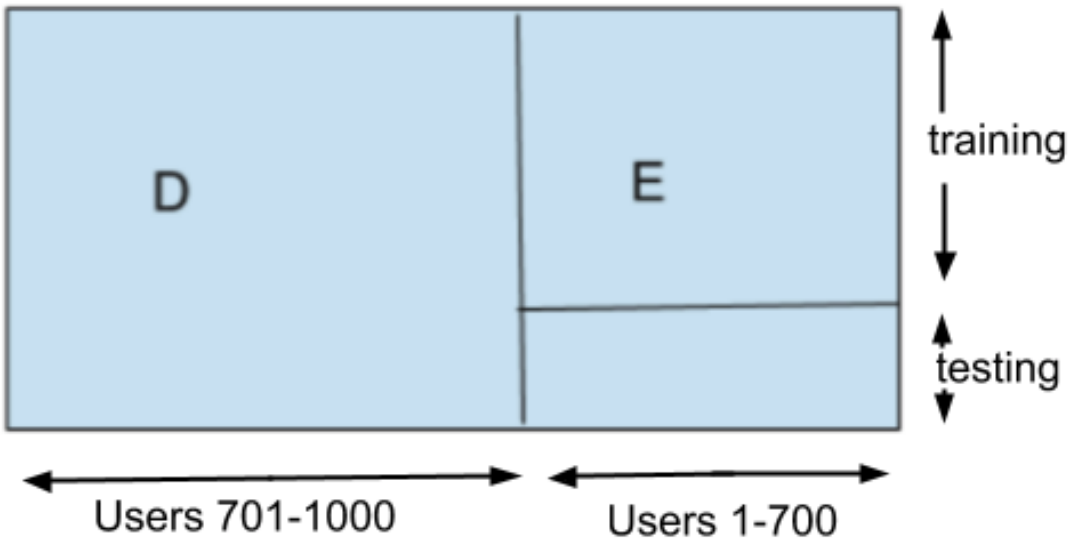


Figure 5.1: Splitting GPDSSyntheticSignature dataset for WI, WD signature verification

Architecture of the model and experimental details:

Model was trained with 'categorical cross entropy' loss and 'rmsprop' optimizer.Experiments were also done with GPDS [10] features. For binary classifiers we experimented with 14, 10 genuine, 14 random forgeries while training, and 10 geniune and 30 skilled forgeries for testing. If X out of 10 are miss-classified, then FRR=X/10, if Y out of 30 are miss-classified, then FAR=Y/30.

Table 5.1: CNN model architecture

Layer name	output shape	pad	stride
Input	1x150x220	-	
Conv2D	96x11x11	0	4
Maxpool	96x3x3	-	2
Conv2D	256x5x5	-	2
Maxpool	256x3x3	-	
Conv2D	348x3x3	1	1
Conv2D	348x3x3	1	1
Conv2D	256x3x3	1	1
Maxpool	256x3x3	-	2
fc	2048	-	-
fc	2048	-	-
fc+softmax	M	-	-

Table 5.2: Classification results on GPDS960 and GPDSSyntheticSignature Dataset

Dataset	samples in E	samples in D	WD Classifier	FRR	FAR
GPDS960	300	581	SVM(linear)	2.52	15.18
GPDS960	300	581	SVM(rbf)	2.26	14.06
GPDSSyntheticSignature Dataset	300	700	SVM(linear)	3.01	15.98
GPDSSyntheticSignature Dataset	300	700	SVM(rbf)	6.83	16.87

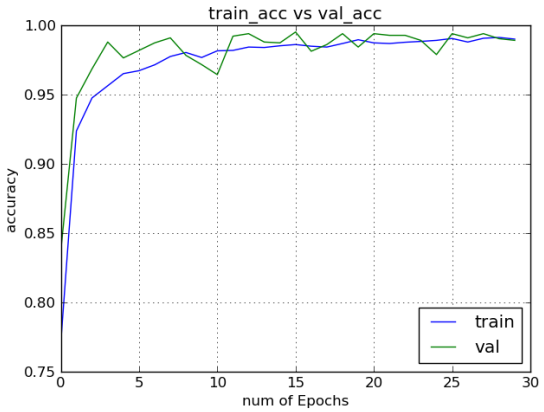


Figure 5.2: CNN training

5.1 WI feature learning using pretrained models

5.1.1 VggNet and ResNet

For training WI system we used pretrained model vgg16 and Resnet50, with GPDSSyntheticSignatue dataset where each user was treated as separate class, we tried training classifier only and also last few layers with softmax layer at end having nodes equal to the number of users in the development set. For ResNet we added:Global average pooling layer, fc of 512 nodes, dropout -0.5, fc of 256 nodes, dropout -0.5, softmax. For VggNet we tried with: after 'block5_pool' fc of 512 nodes, fc of 512 nodes, softmax with 'categorical cross entropy' loss and 'adadelta' optimizer.

Table 5.3: Classification results with pretrained models

Model	acc	WI trainig	epochs	WD Classifier	FRR	FAR
vgg16	76.82%	only genuine	60	SVM(linear)	33.62	9.66
vgg16	-	only genuine	60	SVM(rbf)	26.84	11.33
ResNet50	61%	only genuine	60	SVM(liner)	71.09	4.58
ResNet50	-	only genuine	60	SVM(rbf)	61.32	7.33

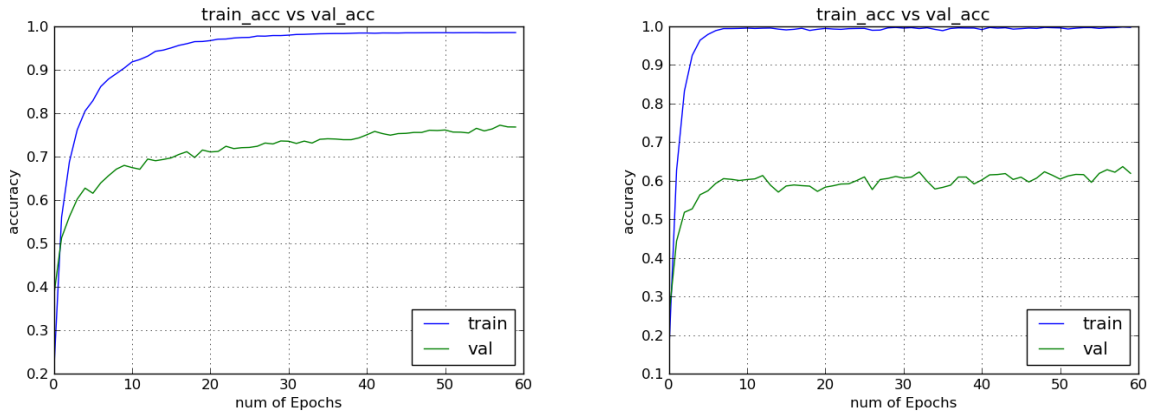


Figure 5.3: trainig and validation loss v/s no. of epochs for ResNet50

5.2 WI feature learning with DCGAN

The architecture of the generator and discriminator used are:

For getting the representation $\phi(X)$ of signature image X we resize it to (152x220) and give it as input to the discriminator and get feature vector of size 1x4480 as output. We tried training model with different no of steps, after 150 steps our generator starts learning

Table 5.4: DCGAN Generator

Layer name	Output shape	parameteres
Dense	267520	27019520
Batchnorm	267520	1070080
Reshape	38x55x128	0
Dropout	38x55x128	0
Upsampling	76x110x128	0
Conv2DTranspose	76x110x128	204864
BatchNorm	76x110x128	256
Upsampling	152x220x64	0
Conv2DTranspose	152x220x64	102464
BatchNorm	152x220x64	256
Conv2DTranspose	152x220x32	51232
BatchNorm	152x220x64	128
Conv2DTranspose	152x220x1	801
Activation	152x220x1	0

Table 5.5: DCGAN Discriminator

Layer name	Output shape	parameteres
Conv2D	76x110x32	832
LeakyRelu	76x110x32	0
Dropout	76x110x32	0
Conv2D	38x55x32	25632
LeakyRelu	38x55x32	0
Dropout	38x55x32	0
Conv2D	19x28x32	25632
LeakyRelu	19x28x32	0
Dropout	19x28x32	0
Conv2D	10x14x32	25632
Flatten	4480	0
Dense	1	4481
Activation	1	0

generating similar images as shown. For getting feature vector for given image we give it as input to discriminator and extract feature vector of 1x4480 from its fully connected layer.

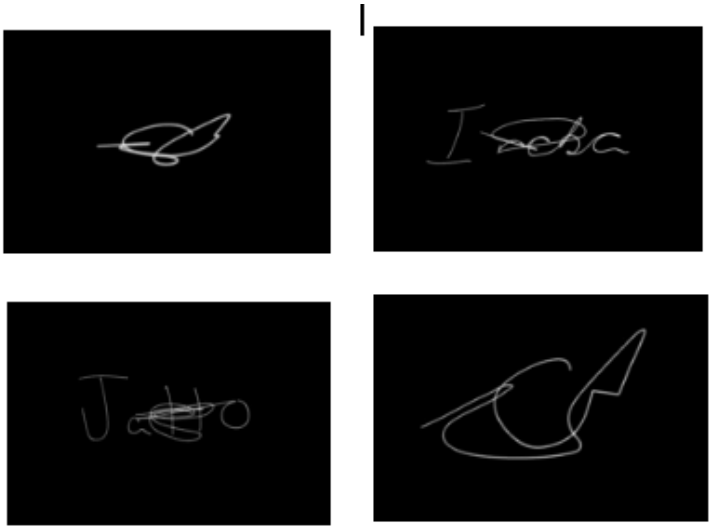


Figure 5.4: Original images

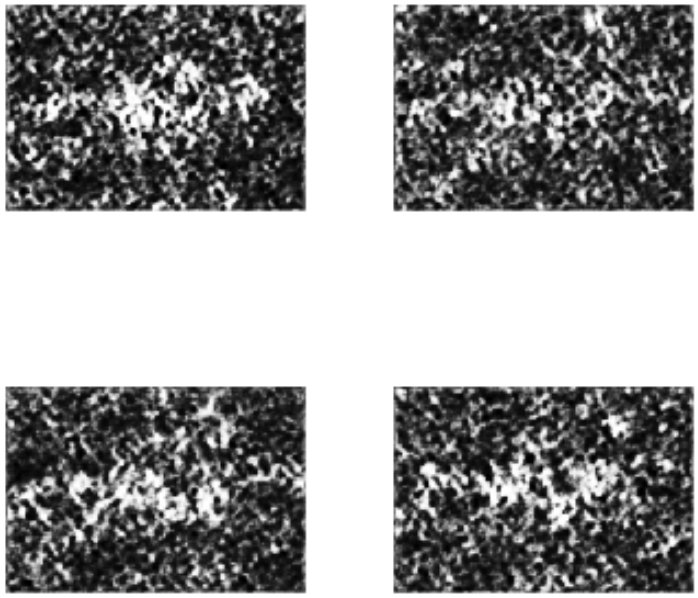


Figure 5.5: Images generated after 80 steps

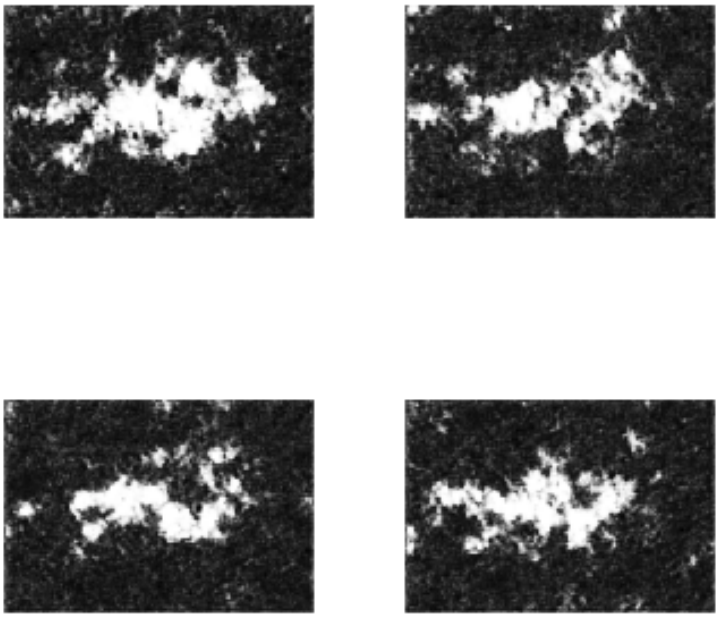


Figure 5.6: Images generated after 120 steps

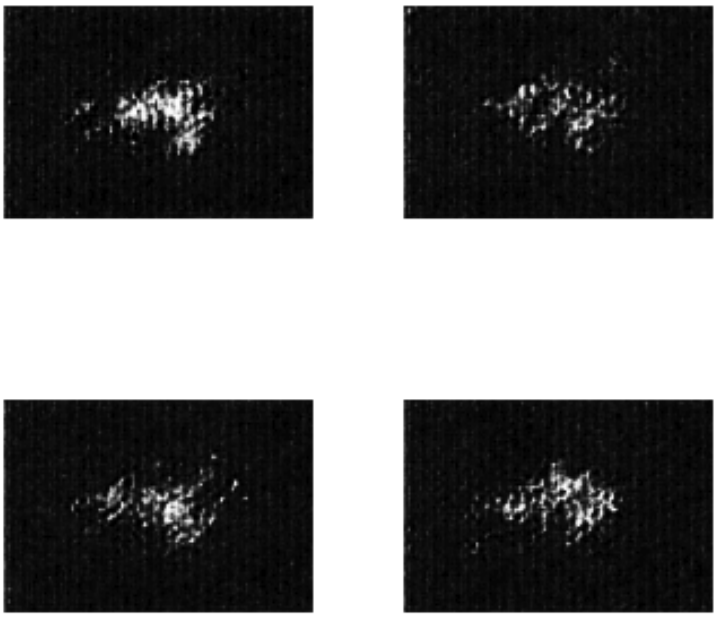


Figure 5.7: Images generated after 160 steps

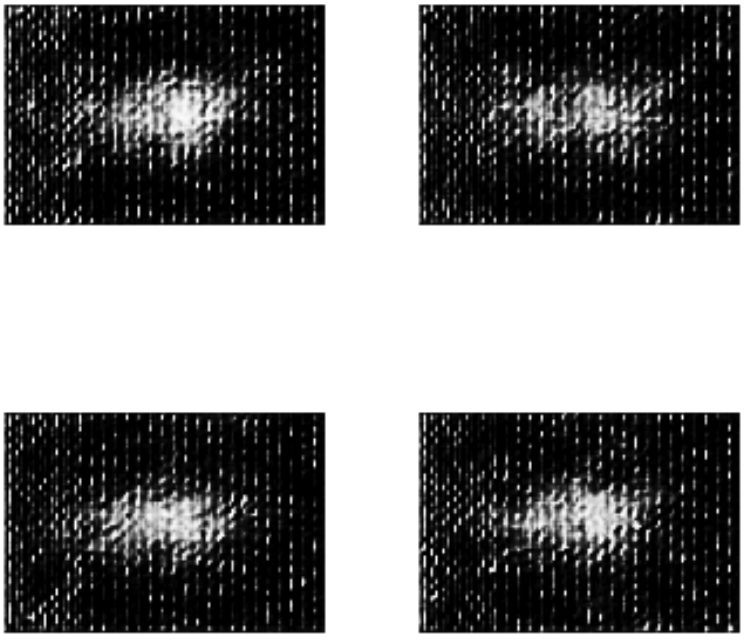


Figure 5.8: Images generated after 200 steps

5.3 WD binary classifier

A binary classifier is trained for each user in E using the feature vectors for images obtained by the trained CNN above with the D, which is used for giving final decision for the query signature as accepted or rejected. We trained SVM with taking different numbers of the genuine signature fron E dataset and with genuine signatures from D dataset (random forgeries) and during testing we used both genuines and forgeries,for a perticular user. Initially for getting best values of hyperparameters γ and C grid search was performed, optimal values are C=1 and γ=2^{-10}.

For Adaboost classifier as WD systems, we are first fitting a simple decision tree with depth=1 as base estimator and in each iteration we are fitting M=5 classifiers with 3 iteration for each user and finally we are calculating average FAR and FRR.

We have tried training WD system with different number (1, 10, 14) of genuine samples here the results are with 14 genuine samples, FAR, FRR decreases with more number of genuine samples.

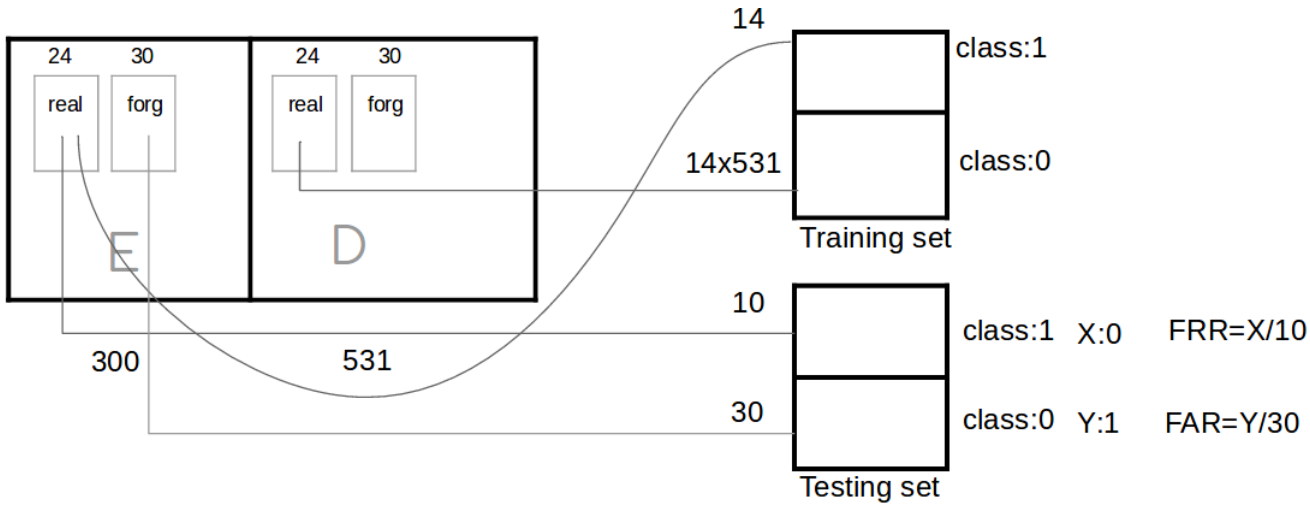


Figure 5.9: WD System training and testing

Table 5.6: GPDS960 and GPDSSyntheticSignature classification comparisons

Dataset	WI system	WD system	FRR	FAR
GPDS960	CNN	SVM(linear)	2.52	15.18
GPDS960	CNN	SVM(rbf)	2.26	14.06
GPDS960	CNN	Adaboost	3.85	11.28
GPDSSyntheticSignature	$CNN_{pretrained}$	SVM(linear)	26.84	11.33
GPDSSyntheticSignature	DCGAN	SVM (linear)	31.78	27.98

5.4 Conclusion and future scope

Here we presented system for signature verification using WI and WD systems, we seen that when signature features are extracted using CNN and Adaboost is used for WD classification we get least FAR and least FRR with SVM as WD. As a future work we can work on unsupervised methods for designing the feature learner because we can get unlimited data for experimenting with. We can generalize the trained WI systems on other publicly available datases like MYCT, ICDAR, Brazilian (PUC-PR). For leveraging benefit of pretrained models will have to do more experiments. We need to improve the performance of WD systems when very less number of samples are available per user.

Bibliography

[1] Vargas, J. F., et al. "Off-line signature verification based on gray level information using wavelet transform and texture features." Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on. IEEE, 2010.

[2] Shekar, B. H., et al. "Grid structured morphological pattern spectrum for off-line signature verification." Biometrics (ICB), 2015 International Conference on. IEEE, 2015.

[3] Khalifa, O., M. K. Alam, and A. H. Abdalla. "An Evaluation on Offline Signature Verification using Artificial Neural Network Approach." 2013 International Conference On Computing, Electrical And Electronic Engineering (Icceee).

[4] Bajaj, Reena, and Santanu Chaudhury. "Signature verification using multiple neural classifiers." Pattern recognition 30.1 (1997): 1-7.

[5] Karanjkar, Shubhangi L., and P. N. Vasambekar. "Signature recognition on bank cheques using ANN." Electrical and Computer Engineering (WIECON-ECE), 2016 IEEE International WIE Conference on. IEEE, 2016.

[6]Chauhan, Prachi, Subhash Chandra, and Sushila Maheshkar. "Static digital signature recognition and verification using neural networks." Information Processing (IICIP), 2016 1st India International Conference on. IEEE, 2016.

[7] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127

[8] Alvarez, Gabe, Blue Sheffer, and Morgan Bryant. Offline Signature Verification with Convolutional Neural Networks. Tech. rep., Stanford University, Stanford, 2016.

[9] Ribeiro, Bernardete, et al. "Deep learning networks for off-line handwritten signature recognition." Iberoamerican Congress on Pattern Recognition. Springer, Berlin, Heidelberg, 2011.

[10] Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Learning features for offline handwritten signature verification using deep convolutional neural networks." Pattern Recognition 70 (2017): 163-176.

[11] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[12] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[13] Zhang, Zehua, Xiangqian Liu, and Yan Cui. "Multi-phase offline signature verification system using deep convolutional generative adversarial networks." Computational Intelligence and Design (ISCID), 2016 9th International Symposium on. Vol. 2. IEEE, 2016.

[14] Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Writer-independent feature learning for offline signature verification using deep convolutional neural networks." Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016.

[15] Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Offline handwritten signature verification—literature review." Image Processing Theory, Tools and Applications (IPTA), 2017 Seventh International Conference on. IEEE, 2017.

[16] http://cs231n.github.io/convolutional-networks/

[17] https://en.wikipedia.org/wiki/Convolutional_neural_network

[18] https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0

LIST OF PAPERS BASED ON THESIS

1. Authors.... Title... *Journal*, Volume, Page, (year).