Writer-dependent Off-line Signature Verification with Neural Networks

Kaihang Fu
Department of Electrical Engineering
Iowa State University
Ames, United States
fkh2016@iastate.edu

***Abstract*—Nowadays along with a growing need for personal authentication in many applications, handwritten signature verification plays an important role in an enormous number of fields such as banks, passport verification system, and documents which are being authorized via signature. Therefore an automatic off-line signature verification is required. This paper presents a novel writer-dependent off-line signature verification system based on a novel set of features and neural network classifier. The proposed system is implemented and tested on ICDAR 2011 SigComp database[12] and GPDS corpus[13]. The experiment results show the verification accuracy rate of 93.3%.**

Keywords—Handwritten signature, Off-line signature verification, Neural Network, Feature extraction

I. INTRODUCTION

Biometric recognition is used in a variety of applications such as secure access to buildings, computer system and ATMs. Such systems aim to recognize an individual based on their physiological (i.e., fingerprint and retina patterns) or behavioral (i.e., voice and handwritten signature) characteristics[1]. By using biometrics, it is possible to confirm an individual's identity requesting their service. Handwritten signatures are widely used as a means to verify an individual's identity for legal purpose on documents such as contract, bank checks, credit card, etc. signature verification is one of the most extensive use method because the process to collect handwritten signatures is quiet easy and non-invasive[2].

Depending on the acquisition method, signature verification systems are divided into two categories: online (dynamic) and offline (static). In the online case, the signature is acquired by an acquisition device such as a tablet, and the dynamic information of the signature is captured (position of the pen, pen inclination, pressure, etc.). In the offline case, the signature acquired after the writing process is completed and represented as a digital image[3]. In this paper, we focus on the offline (static) signature verification problem.

During last few years, some efforts have been already made by different researchers in offline signature verification area. Jaiswal and Kasetwar presented an offline signature verification system using a feature set with neural networks[4]; AI-Maqaleh and Musleh proposed a novel feature of extraction method based on static image splitting[5]; Vahid et al. presented a novel method for extracting easily computed rotation and scale invariant features for offline signature verification[6]; Ramachandra et al. proposed a robust offline signature verification based on global features[7]; B. Fang et al. proposed two method to track the signature variations[8]; Shanker and Rajagopalan proposed a signature verification system based on Dynamic Time Warping (DTW)[9].

Offline signature verification problem can be performed in either of the two different approaches: writer-independent (WI) and writer-dependent (WD). The former approach uses a classifier to match each input questioned signature to reference signatures, and a single classifier is trained for all writers, the most important advantage of WI approach is that the classifier is trained only once, so this system is economic. However, the WI approach has many disadvantages: it needs a lots of sample signatures to training, and its accuracy of verification is not good. Rivard et al.[10] and Kumar et al.[11] proposed their different writer-independent signature verification system. For the WD approach used in this paper, the system models the signature of each individual from his samples, and a specialized classifier is trained for this particular writer. The WD system needs much fewer sample signatures for training a classifier, and more importantly, the WD system is more accurate.

In this paper we proposed a novel writer-dependent signature verification process which only needs very fewer genuine signatures (i.e. 24 genuine signatures) and forgeries (i.e. 13 forgeries) for training of classifier. Another contribution of this work is that we propose a novel feature set for offline signature verification. Our experiment results show really high verification accuracy. In this work, we consider only skilled forgery, not random forgery, not simple forgery.

Rest of the paper is as follows: Section II describes the signature verification system scheme. Section III, IV&V present the preprocessing, feature extraction & pairing and classification stage respectively. Section VI demonstrates the verification process while Section VII presents experiment results along with the experimental setup and the description of databases used. The conclusion and the scope of future work are reported in Section VIII.

II. VERIFICATION SYSTEM

My proposed signature verification system is described in Fig. 1.The inputs of the system are two signature images, one is genuine and the other is questioned signature to be tested. The two input images are first given to preprocessing unit where several preprocessing operations such as cropping, binarization, resizing etc. are performed. Then from each preprocessed image, various features which can distinguish writers of signatures are combined in pairing unit, I use one of the simplest ways—absolute difference of

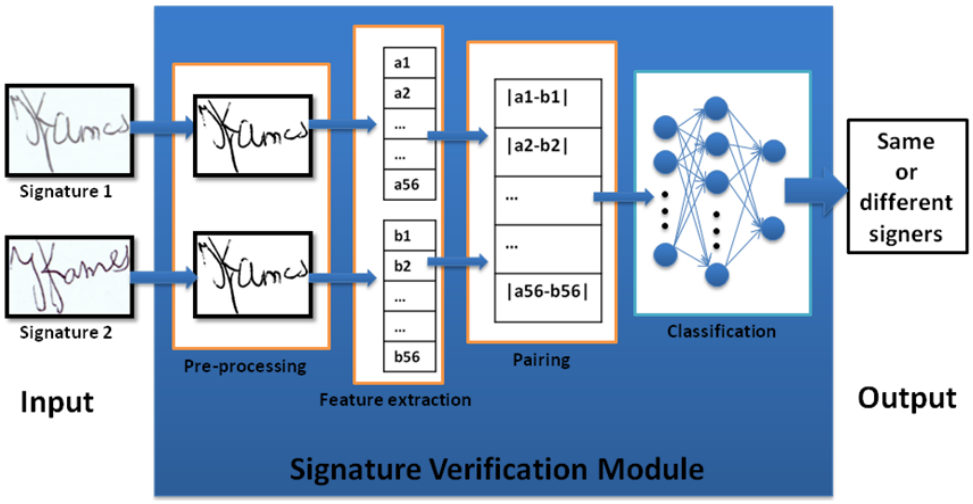

Fig. 1. A schematic diagram of the proposed signature verification system

corresponding element of the feature vectors as the pairing vector in this paper. Then the pairing vector is fed into a two-class classifier, which is a trained neural network. Different writers have different trained neural network classifiers because the verification system is writer-dependent. Finally, this signature verification module produce the output whether the two signatures are written by the same signers or different signers.

Next, preprocessing, feature extraction, pairing and classification will be discussed in detail.

III. Preprocessing

Preprocessing is an essential stage for off-line signature images analysis. Scanned signature images can be provided by many on-line databases (such as *GPDS corpus* and *ICDAR SigComp database* being used in this paper). These raw signature images have to be processed with many typical preprocessing operations to improve and enhance the signature images, so as more accurate information can be extracted as features of signature[4]. In this paper, I use *MATLAB* to preprocess the signature images.

A. Loading the Image and Conversion to Grayscale

Load the signature image into *MATLAB* with the "imread()" command. If the image is colored (such as images in *ICDAR SigComp database*), it is stored as a three dimensional of array corresponds to intensity level of Red-Green-Blue (RGB). Then convert the RGB image of 3-D array to a grayscale image of 2-D array with intensities from 0 to 255. The *rgb2gray()* command was used.

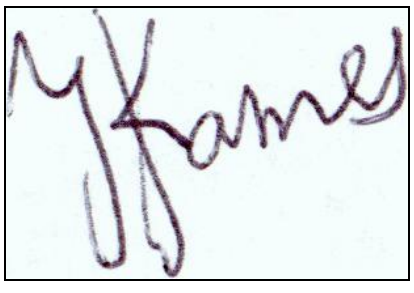

Fig. 2. Grayscale conversion of Image

B. Cropping the Image

The signature image contains some unnecessary space that is not used during the verification process[5]. Therefore, we need to crop the image before feature extraction.

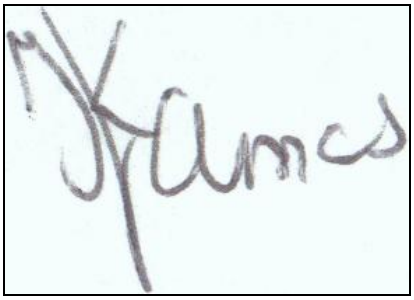

Fig. 3. Grayscale conversion of Image

C. Binarization

The next step binarization is a process which convert the grayscale image to binary image. As shown in Fig. 4, the signature is black (value is 0) and the background is white (1). The *im2bw()* in *MATLAB* was used in this step.

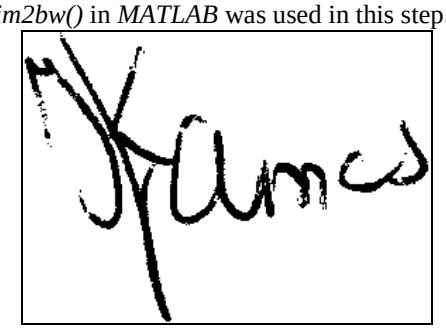

Fig. 4. Binarized image

D. Resizing the Image

Scanned image can be any size, so we need to resize the loaded image to a standard size so as to let the verification system is independent of the size. Here, I defined the standard size is 256×256 pixels, as shown in Fig. 5. The *imresize()* command in *MATLAB* was used in this step.

Fig. 5. Resized image in standard 256×256 pixels

E. Thinning(Skeletonization)

Thinning the image means removing some signature pixels from the binary image, which making the signature image one pixel thick. The goal of Thinning is to eliminate the thickness difference of pen so as to make the verification system independent of signature width. Thinned image is shown in Fig. 6. The *bwmorph()* command in *MATLAB* was used in this step.

Fig. 6. Thinned signature image

IV. FEATURE EXTRACTION

Extracting features that distinguish between genuine and forged signatures is the most challenging and important step in automatic signature verification[5]. In this paper, the features extracted can be of two types as global features or local features, they are both parameter features. Global features are extracted from whole signature image and represent information of entire signature. Local feature are extracted from a part of signature image. For instance, whole signature image can be divided into 16 grids, and local features can be extracted from each grid. There is no a standard feature that can validate authenticity of any signature written by different persons. Feature selection and extraction play a vital role in signature verification, extracted features should contain essential discriminative information and represent it in an appropriate form which can beefed into a classifier[4]. Excellent feature set can improve the accuracy of the verification process. In this paper, I used a novel feature set including 8 global features and 48 local features. The details of these features are as follows.

A. Global Features

1) Number of signature pixels (in thinned image)

The first feature which can be extracted is the number of black pixels in thinned signature image of 256×256 pixels size. This feature also can be regard as the trajectory length of a signature, and any two genuine signatures written by one person are nearly same. This feature can be extracted by simple codes in *MATLAB*:

```
k=1;
for i1=1:256
   for j1=1:256
         if(I3(i1,j1)==0)
                  k=k+1;
               end
      end
end
N = k-1;
```

The value N is the total number of pixels in thinned signature.

2) Area of the signature (in resized image)

Area of the signature is equal to total number of pixels in resized binary signature before thinning. This feature can be extracted by the similar codes as the previous feature in *MATLAB*, only the applied image should be changed to binary image.

```
a=1;
for i5=1:256
   for j5=1:256
         if(I2(i5,j5)==0)
                  a=a+1;
               end
      end
end
area = a-1;
```

3) Aspect ratio (in binary image)

The aspect ratio is defined as the ratio of width to height of the signature bounding box, which varies for different individuals[6]. The binary signature image had been cropped, so aspect ratio can be directly gotten by calculating the ratio of width to height of the binary image.

$$\text{aspect}_{\text{ratio}} = \frac{\text{Width}}{\text{Height}} \tag{1}$$

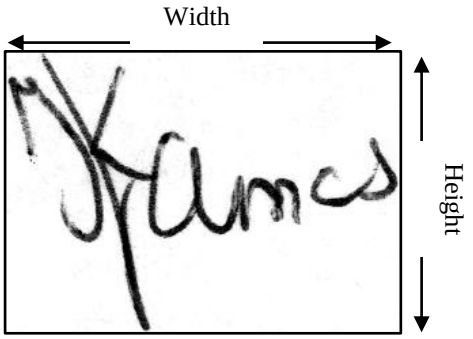

Fig. 7. Signature bounding box

4) The inclination angle of the line joining the center of gravity and the lower right corner (in binary image)

This feature is the angle between the line connecting the center of gravity to the lower right corner of the bounding box and the horizontal axis. The way to calculate gravity center is given in (2).

$$\begin{cases} X_g = \frac{\sum_{i=1}^{N} x_i}{N} \\ Y_g = \frac{\sum_{i=1}^{N} y_i}{N} \end{cases} \tag{2}$$

N is the total number of signature pixels in binary image. x_i is the value of X co-ordinate of i-th pixel, y_i is the value of Y co-ordinate.

The angle is calculated by (3).

$$\text{Angle} = \tan^{-1}\left(\frac{m - Y_g}{n - X_g}\right) \quad (3)$$

m is the width of the bounding box, n is the length.

5) Vertical variance & Horizontal variance (in binary image)

These two features represent the departure of vertical &horizontal co-ordinate values from the gravity center. They can be calculated by (4).

$$\begin{cases} H\_var = \sum_{i=1}^{N}\left(\frac{x_i - X_g}{n}\right)^2 \\ V\_var = \sum_{i=1}^{N}\left(\frac{y_i - Y_g}{m}\right)^2 \end{cases} \quad (4)$$

6) Intersection points & Border points (in thinned image)

Extracting these two features is to count the number of special pixels in thinned signature image. Border point represents the start or end of a stroke belonging to signature image. In thinned signature image, border point is the pixel which has only 1 neighbor equal to "0" in total 8 neighbors[7]. Intersection point represents the union of two or more lines belonging to signature image. An intersection point is a pixel point which has more than 3 neighboring pixels equal to "0" in 8 neighbors[4].

B. Local Features

Local features extraction process is based on adaptive signature image partitioning. The preprocessed signature image is split into 4 parts through its center of gravity. Formula (2) is used to calculate center of gravity. Then each part is partitioned 4 equal parts. So the signature image are divided into 16 sub-image cells. Some local features are extracted from each cell[6]. As shown in Fig. 8 and Fig. 9.

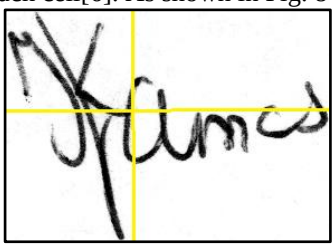

Fig. 8. Image is split into 4 parts through its center of gravity

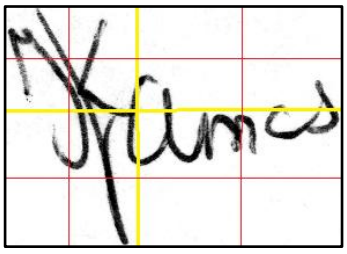

Fig. 9. Image is divided into 16 sub-image cells

1) Pixel density

a) To count the total number of pixels (area of cell) in each cell;

b) To count the total number of signature pixels (black pixels) in each cell;

c) The pixel density is calculatedwith (5).

$$\text{pixel density} = \frac{\text{Total number of signature pixels in each cell}}{\text{The area of each cell}} \quad (5)$$

2) Pixel angle

a) To calculate the inclination angle of each line joining each signature pixel (black pixel) and the lower right corner of the cell;

b) To calculate the sum of pixel inclination angles in each cell;

c) The pixel angle is computed with (6).

$$\text{pixel angle} = \frac{\text{Angle sum in each cell}}{\text{Total number of signature pixels in each cell}} \quad (6)$$

3) Pixel distance

a) To calculate the distance between every signature pixel (black pixel) in each cell and the lower right corner of the cell;

b) To calculate the sum of distances in each cell;

c) The pixel angle is computed with (7).

$$\text{pixel angle} = \frac{\text{Distance sum in each cell}}{\text{Total number of signature pixels in each cell}} \quad (7)$$

Three kinds of local features were extracted from each cell, so totally 48 local features can be extracted from one signature image.

V. Pairing and Classification

A. Pairing

A 56-value feature vector can be extracted from one signature image, including 8 global features and 48 local features. So two 56-value feature vector were extracted from one genuine signature and one questioned signature, then they are combined in pairing unit. There some different pairing method, here I used a simple way----absolute difference of corresponding element of the feature vectors as pairing vector[11]. If original feature vectors are $[f_1^1, f_2^1, \ldots\ldots, f_{56}^1]$ and $[f_1^2, f_2^2, \ldots\ldots, f_{56}^2]$, the pairing vector is $[|f_1^1 - f_1^2|,\ |f_2^1 - f_2^2|, \ldots\ldots, |f_{56}^1 - f_{56}^2|]$.

This pairing vector will be fed into a two-class neural network classifier.

B. Classification

After features are extracted, a two-class classifier needs to be built into verify the authenticity of the signature. Many machine learning techniques can be employed to do classification, such as Neural Network (NN), Support Vector Machine (SVM), or a threshold decision method. Neural Network can model any complex functions and its excellent performance in pattern recognition make NN be used widely.

1) NN architecture

The architecture of an NN is referred to as the interconnection structure between different layers of neurons[6]. In this proposed system a three-layer Neural Network was employed for verification task. Three layers are: the input layer, hidden layer and the output layer.

The number of hidden layer would influence the performance and calculation cost of NN classifier. In most of the time using more layers of hidden neurons could increase the accuracy and calculation cost, because deeper layers and more hidden neurons could enable Neural Network to model more complex function to classify data. However, as the network complexity increases, the possibility of the network overfitting the training data and calculation cost will increase. According to my experience on layer of hidden layer in this system is best, the number of hidden neurons is set to be 256. As shown in Fig. 10.

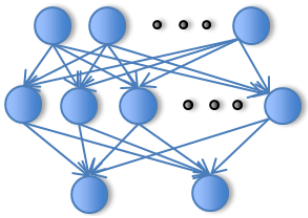

Fig. 10. Neural Network structure

2) Training NN

Neural network training is a supervised learning process. A lot of feature pairing vectors and labels are fed into NN as train data, the NN can learn an unknown input-output relationship to solve signature verification problem. After the training, some others pairing and labels are fed into NN as test data, then calculating the accuracy to judge the performance of NN classifier. I used *Keras*[14]library to build and train Neural Network in this paper. *Keras* is a high-level neural network library written in *Python* and capable of running on top of *TensorFlow*, *CNTK* or *Theano.* In order to improve the performance of NN, I standardize input pairing vector, add dropout layer, Gaussian noise layer and L2 regularization to NN. The parameter of Neural Network is shown in TABLE I. .

TABLE I. PARAMETERS OF NN BUILT IN *KERAS*

PARAMETER	VALUE
Gaussian noise layer standard deviation	0.02
# of input neurons	56
# of neurons in hidden layer	256
L2 regularizer penalty	0.02
Hidden layer activation	relu
Dropout rate	0.5
# of output neurons	2
Output layer activation	Softmax
Optimizer	adam
Learning rate	0.002
Decay	0.005
Batch size	100
Callback	Earlystopping

VI. VERIFICATION PROCESS

The verification process is described in Fig. 11. In actual case, Forensic Handwriting Examiners (FHEs) often need to examine the authenticity of signatures in agreement or others document. Imagine we have a questioned signature and a signer, we need to judge whether the signature was written by the signer. Firstly let the signer writes N (such as 10 or 20) genuine signatures, and we produce M (such as 20 or 30) forged signatures by others people or computer program. Alex Graves of University of Toronto described how Long Short-term Memory recurrent neural network (LSTM RNN) can be used to generate forgeries according to on-line handwriting[15].

After obtain a certain number of genuine and forged signatures, the pairing module could produce a lots of pairing vectors, including genuine-genuine pairs and genuine-forgery pairs. These pairing vectors are train set which is used to train a neural network classifier belongs to this signer. Now we can use the trained NN classifier to judge if the questioned signature is written by the signer.

The questioned signature and N genuine signatures written by the signer are fed into the signatures verification module, after pre-processing, feature extraction and pairing, N genuine–questioned pairing vectors are fed into trained NN classifier. Each genuine-questioned pairing vector serves as a vote. The two-class classifier would give us N results, including the same signers (1) or different signers (0). If the classifier deems that the questioned signature differs from too many of genuine signature, it is classified as a forgery. In other words, if the number of result *1* is more than result *0*, we can conclude that the questioned signature is genuine (written by the signer); if the number of result *0* is more, then the conclusion is that the questioned signature is a forgery.

VII. EXPERIMENT RESULTS

Experiments are based on two widely used public databases, namely *GPDS synthetic Signature Corpus* and *ICDAR 2011 SigComp database*. Description of experimental data design and results are as follows.

A. ICDAR 2011 SigComp Database

This database comes from the International Conference on Document Analysis and Recognition (ICDAR) 2011 SigComp international signature verification[12]. The database includes both online and offline signatures for both Dutch and Chinese signers, and is split into training set and testing set. In this paper I only use the offline Dutch training set. The Dutch training set contains a total of 361 signatures and 12 forged signature for each signer. The 12 forged signatures of each signer come from 3 others forgery writers, each one writes 4 forged signatures, every forgery is skilled forgery.

Since this signature verification system is writer dependent, for each signer I divide signatures belong to him into a training set and testing set. Each training set comprises of 22 genuine signatures and 8 forged signatures,

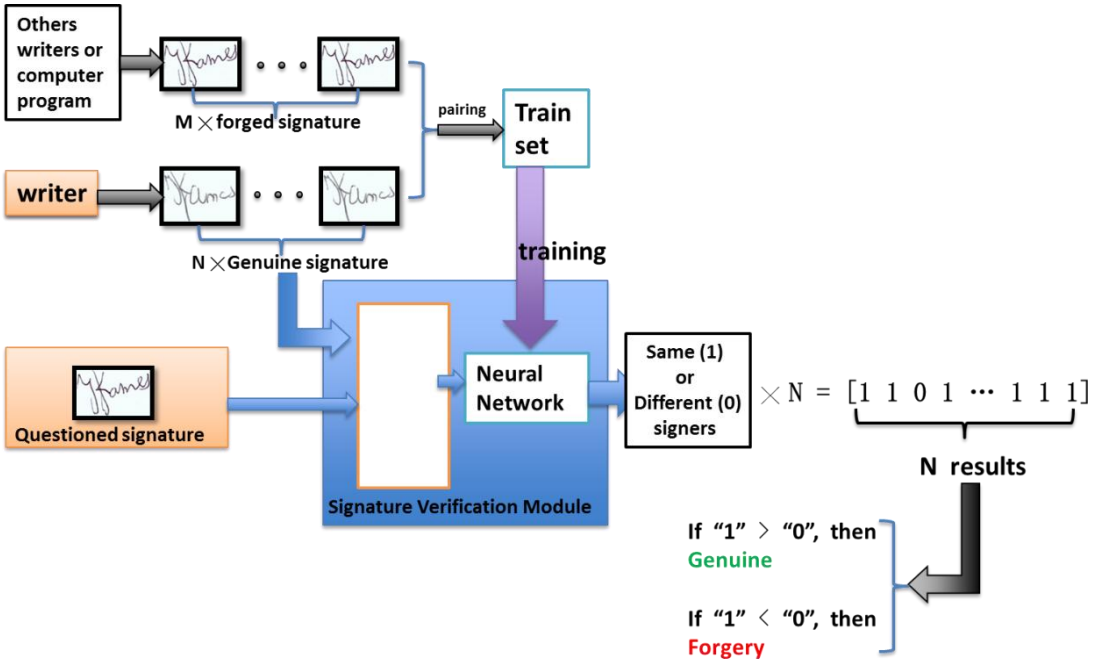

Fig. 11. Verification process

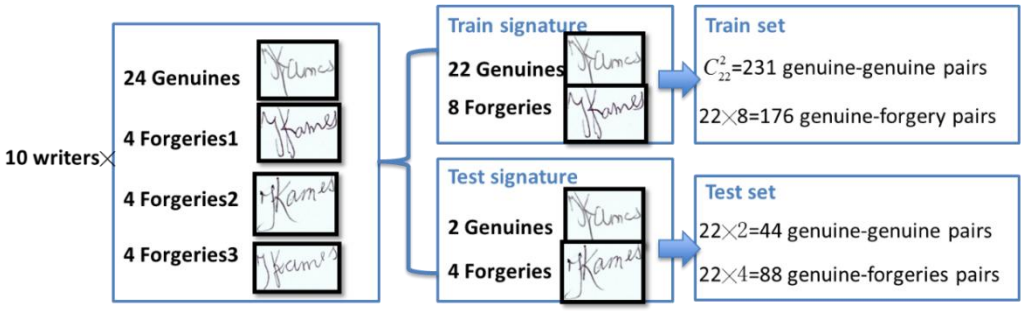

Fig. 12. Dividing *ICDAR 2011 SigComp Database*

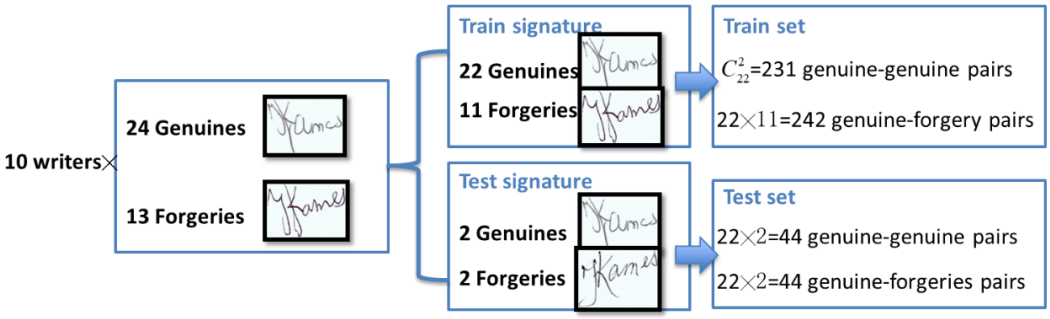

Fig. 13. Dividing *GPDS Corpus*

each testing se comprises of 2 genuine signatures and 4 forged signatures. Since 22 genuine signatures and 8 forged signatures in training set is respectively regard as N genuine signatures (N=22) written by the signer and M forged signatures (M=8) produce by examiners in 6-th chapter *Verification Process*, we get $C_{22}^2 = 231$ genuine-genuine pairs of signatures and $C_{22}^1 \times C_8^1 = 22 \times 8 = 176$ genuine-forged pairs of signatures for training set of each signer. Totally 231+176=407 pairing vectors would be used to train a special NN classifier solely for this signer. Since 2 genuine and 4 forged signatures in testing set are all regard as questioned signatures to be tested in 6-th chapter *Verification Process*, every questioned signature needs to be paired with genuine signatures written by the signer, we get

$22 \times (2+4) = 132$ pairs of signatures. Each questioned signature would be classified by the trained NN according to those genuine-questioned pairs' verification results.

To simulate different case in real life, I design 2 different experiment based on different dividing ways to make training and testing set, the unseen forgery-writer test and all forgery-writer test. Details would be discussed in following section.

B. GPDS synthetic Signature Corpus

This database contains signature image from 10000 synthetic individuals: 24 genuine signatures and 30 forgeries for each individual, all forgeries are skilled[13]. All the offline signatures were generated with different modeled pens, and all signatures images in this corpus are in binary form.

Since the database is very large, I only use first 100 individuals' signatures to do experiment. From signatures belong to each individual, I randomly select 22 genuine signatures and 11 forged as training set, and 2 genuine signatures and 2 forged signatures as testing set. Therefore, we can get $C_{22}^2 = 231$ genuine-genuine pairs of signatures and $C_{22}^1 \times C_{11}^1 = 22 \times 11 = 242$ genuine-forged pairs of signatures for training set of each individuals. This training set contains 231+242=473 pairing vectors. For each individual's testing set, we can get $22 \times (2+2) = 88$ pairs of signatures, 4 questioned signatures in testing set would be classified by their respective NN classifier which are trained with the training set.

C. Experiment Results

1) Evaluation Metrics

Three different types of parameters have been used to measure the performance of a signature verification system. These are False Acceptance Rate (FAR), False Rejection Rate (FRR) and Accuracy[5].

FAR concerns with the false rejection of genuine signature. FRR concerns with the false acceptance of forged signature. Accuracy measures how many signatures are correctly classified. Their mathematical calculation formulas is shown as follows.

$$\text{FAR} = \frac{\text{Number of forgeries accepted}}{\text{Number of forgeries tested}} \tag{8}$$

$$\text{FAR} = \frac{\text{Number of genuine signatures rejected}}{\text{Number of genuine signatur es tested}} \tag{9}$$

$$\text{FAR} = \frac{\text{Number of signatures classified correctly}}{\text{Number of signatures tested}} \tag{10}$$

2) ICDAR 2011 Database Test Results

a) Unseen Forgery-writer Test

This test is to test the classifier on a testing set where forgeries are written by entirely different people than the training set. In actual case, examiner need to let others people or computer program produce forgeries in training set, if the questioned signature is a forgery, then it comes from a different individual. This test aims to simulate this kind of actual cases. Therefore, we need to let forgeries in training set and forgeries in testing set are written by different people. We know the 12 forgeries of each individual in database come from 3 different person, each one writes 4 skilled forgery. So we assign 4 forgeries written by one person to testing set, the other 8 forgeries to training set.

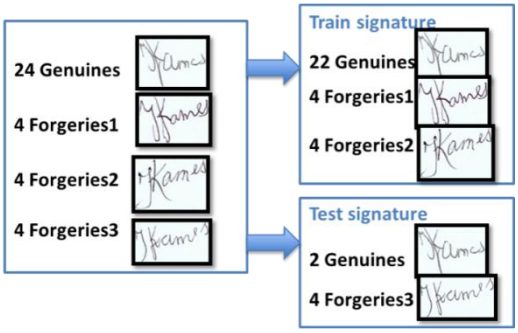

Fig. 14. Unseen Forgery-writer Test dividing way

When we select the forgery-writer whose forgeries are assign to testing set, we can select randomly, or we can select the best forgery-writer whose forgeries are most similar to genuine signatures, or the worst one, or the middle level one. The different test results are reported in TABLE II. .

TABLE II. PERFORMANCE OF UNSEEN FORGERY-WRITER TEST

	F_random	F_worst	F_middle	F_best
FAR	**10%=4/40**	5%=2/40	15%=6/40	22.5%=9/40
FRR	**0%=0/20**	0%=0/20	0%=0/20	0%=0/20
Accuracy	**93.3%=56/60**	96.7%=58/60	90%=54/60	85%=51/60

According to table 2, we can know that the performance on F_worst is best. It is predictable because the worst forgery is easiest to be classified, and the more skilled the forgeries in training set, the better neural network classifier be trained. We also observe that FRR is always 0%, this is because if the questioned signature is genuine, then the questioned signature and genuine signatures in training set are written by the same person, and the neural network had remembered this writer in training learning process. So genuine signature usually can be classified correctly.

b) All Forgery-writer Test

In actual case, if examiners have a suspect signature database and the writer of the questioned signature is included in the database, this database would be the training set of the verification. This test aims to simulate above case, we assign half of signatures of each forgery-writer to testing set, the other part of forgeries are assigned to training set. So the training set and testing set both contains 6 forgeries.

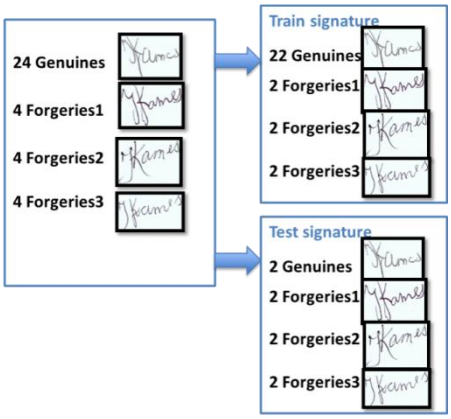

Fig. 15. All Forgery-writer Test dividing way

TABLE III. PERFORMANCE REPORT OF ALL FORGERY-WRITER TEST

	F_all
FAR	6.6%=4/60
FRR	0%=0/20
Accuracy	95%=76/80

We can observe that the accuracy is higher than the unseen forgery-writer test. It is because the neural network had learned all the writers in training process.

3) GPDS Corpus Test Results

We select 4 signatures as questioned signatures for each individual from database, the two are genuine signatures, others two are forged signatures. So we have $4 \times 100 = 400$ signatures to be tested, comprising of 200 genuine signatures and 200 forged signatures. Forged signatures are all skilled.

The performance of proposed system with other existing system tested on GPDS Corpus is reported in TABLE IV. .

TABLE IV. PERFORMANCE REPORT OF PROPOSED SYSTEM WITH EXISTING

	Proposed system	[10]	[5]	[11]
No. of signers	**100**	160	300	300
FAR	**12%**=24/200	5.99%	13.99%	13.76%
FRR	**2%**=4/200	19.81%	13.08%	13.76%
Accuracy	**93%**=372/400	89.30%	86.64%	86.24%

We can observe that the proposed system has the best performance over other three systems.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose a writer-dependent off-line signature verification system based on a novel feature set comprising the combination of global & local features with neural network as classifier. We get test accuracy that higher than 90% in experiment. This system is practical in actual case because it only need a small set of signatures as training data which are easy to collect and it is really fast to finish verification. This proposed system is ingenious, simple and accurate enough.

More training data, more representative feature set and better tuned neural network classifier can be regard as future works to improve the system.

REFERENCES

[1] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometricrecognition," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 14, no. 1, pp. 4–20, 2004.

[2] R. Plamondon and S. N. Srihari, "Online and off-line handwritingrecognition: a comprehensive survey," Pattern Analysis and MachineIntelligence, IEEE Transactions on, vol. 22, no. 1, pp. 63–84, 2000.

[3] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Offline Handwritten Signature Verification-Literature Review," arXiv preprintarXiv:1507.07909, 2015.

[4] Snehil G. Jaiswal, Abhay R Kasetwar, "Off-line Signature Verification Using Global &LocalFeatures with Neural Networks", IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 1525-1531, 2014.

[5] B. M. Al-Maqaleh, A. M. Q. Musleh, "An Efficient Offline Signature Verification System usingLocal Features", International Journal of Computer Applications (0975 – 8887), Volume 131 – No.10, pp. 39-44, Dec. 2015.

[6] V. Malekian, A. Aghaei, M. Rezaeian, M. Alian,"Rapid Off-line Signature Verification Based onSignature Envelope and Adaptive DensityPartitioning", IEEE First Iranian Conferenceon Pattern Recognition and Image Analysis (PRIA), pp. 1-6, 2013.

[7] C. RamachandraA, J. S. Rao, K. B. Raja, K. R. Venugopla and L. M. Patnaik, "Robust Offline Signature Verification Based On Global Features," *2009 IEEE International Advance Computing Conference*, Patiala, 2009, pp. 1173-1178.

[8] B. Fang, C.H. Leung, Y.Y. Tang, K.W. Tse, P.C.K. Kwok, Y.K. Wong, "Off-line signature verification by the tracking of feature and stroke positions", Pattern Recognition, Volume 36, Issue 1, 2003, pp. 91-101.

[9] A. Piyush Shanker, A.N. Rajagopalan, "Off-line signature verification using DTW", Pattern Recognition Letters, Volume 28, Issue 12, 2007, pp. 1407-1414.

[10] D. Rivard, E. Granger, R. Sabourin, "Multi-feature extraction and selection in writer-independent off-line signature verification", International Journal on Document Analysis and Recognition (IJDAR), vol. 16, no. 1, pp. 83-103, 2013.

[11] Rajesh Kumar, J.D. Sharma, Bhabatosh Chanda, "Writer-independent off-line signature verification using surroundedness feature", Pattern Recognition Letters, Volume 33, Issue 3, 2012, pp. 301-308.

[12] http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2011_Signature_Verification_Competition_(SigComp2011)

[13] http://www.gpds.ulpgc.es

[14] https://faroit.github.io/keras-docs/1.2.2/

[15] Alex Graves, "Generating Sequences With Recurrent Neural Networks", arXiv e-printarXiv:1308.0850, 2014