A Deep Learning-Based Offline Signature Verification Method by Single Known Sample

Hsin-Hsiung Kao* and Che-Yen Wen

Department of Forensic Science, Central Police University, Taiwan

ABSTRACT

Signature verification is one of the popular biometric techniques for personal identification. Although automatic signature verification systems have attracted researchers' attention for a long time, there are few attempts to perform the verification based on a single known sample. In this paper, we propose an off-line handwritten signature verification method by using a unique local feature extraction approach and deep convolutional neural network (CNN). In the training process, our CNN is trained by only single genuine (known) and some forged signature samples. In the testing process, the proposed method can verify a questioned signature as genuine or forgery (all questioned signatures and forged authors were not present in the training process). We use the open source dataset, Document Analysis and Recognition (ICDAR) 2011 SigComp in the experiments, and get the accuracy of 98.41%, FAR of 0.91% and FRR of 2.88% in our testing dataset.

Keywords: handwritten signature verification, convolutional neural network (CNN), deep learning, forensic science

基於深度學習及單一已知簽名樣本的離線驗證方法

高信雄*　溫哲彥

中央警察大學鑑識科學學系

摘　　要

簽名驗證為主流的生物特徵識別技術之一，儘管相關研究已行之有年，卻鮮少有學者基於單一已知簽名樣本的驗證問題進行研究。本文以獨特的局部特徵擷取法結合深度卷積神經網路（CNN）提出創新的離線簽名驗證方法。本研究透過單一真實簽名樣本輔以數個偽造簽名進深度學習訓練，於測試過程中，所有未知簽名及偽造簽名的作者皆不曾出現在訓練過程，以確保系統的有效性。本文簽名樣本採自 ICDAR 2011 SigComp 公開資料集，並取得 98.41%的準確率、0.69%的錯誤接受率（FAR）及 2.88%的錯誤拒絕率（FRR）。

關鍵詞：手寫簽名驗證，卷積神經網路(CNN)，深度學習，鑑識科學

文稿收件日期 108.3.18;文稿修正後接受日期 109.5.4; *通訊作者
Manuscript received March 18, 2019; revised May 4, 2020; * Corresponding author

I. INTRODUCTION

Handwritten signature verification plays an important role in biometric authentication. In practice, well-trained experts perform it with visual comparison. In order to increase reliability and efficiency, automatic signature verification methods have been developed for decades. Some of them have applied deep learning methods and shown their capability for verification. Most of these methods require several (more than one) genuine reference signature samples for the feature extraction and model training processes [1-3]. However, it is not easy for police officers to get "enough" reference signature samples for examination in first-line law enforcement. In Taiwan, the forensic document examiners of Criminal Investigation Bureau (CIB) perform most of the handwritten signature examinations for both civil and criminal cases. They will need to collect more signature samples from the suspect if there are too few features in current samples. If the case is still under preliminary investigation, the investigators usually need a quick examination (to evaluate the possibility) to progress their investigation. In this case, they will need a quick verification method with few samples.

Another example is the use of commercial signature verification. When considering customer convenience, some companies may only ask customers to provide a limited number (or even only one) of the signature samples. There are only a few researches [4] attempt to perform signature verification with single known reference signature. And, they are based upon handcrafted feature algorithms and do not consider skilled forgeries (i.e., only for the random forgery problem).

In this paper, we propose an off-line handwritten signature verification method by using a unique local feature extraction approach and convolutional neural network (CNN). The proposed method can classify a questioned handwritten signature as a genuine one or skilled forgery.

The rest of the paper is organized as follows. First, we provide a brief discussion of automatic signature verification and survey related works in Section 2. We introduce the CNN used in the paper, in Section 3. Section 4 explains the design of experiments, including data acquisition and the network architecture used for training. Section 5 shows the training process and evaluates the performance of our model. Finally, we conclude with the advantages and limitations of our approach and describe our future works in Section 6.

Ⅱ. BACKGROUND AND RELATED WORK

2.1 Signature Verification

Handwriting signature verification methods can be classified into two categories: online and offline. In general, offline methods are more challenging than online ones, since we cannot acquire additional dynamic information from the signature tablet or other digital devices. Furthermore, comparing to other biometric technique, handwriting signatures have relatively low inter-class variability (the variability between genuine signatures and skilled forgeries) and high intra-class variability (the variability among an individual's genuine signatures, see Fig. 1). For these reasons, although offline signature verification is widely used for the purpose of personal authentication, it is still one of the most challenging problems in biometrics.

Fig. 1. An example of multiple signatures written by the same person. It shows a high variability between samples. Source: Justino et al., 2000 [5].

2.2 Feature Extraction

The offline handwritten signature verification methods can be classified as two types: handcrafted feature extraction and deep learning methods [3].

The first type of approaches uses the handcrafted feature extraction methods to find feature descriptors for offline signatures. In last few decades, several survey papers [6-9] have reviewed the handcrafted feature extraction and

classification methods. For example, Deng et al. [10] proposed a wavelet-based verification system by curvature features. Drouhard et al. [11] took the signature outline as a global shape feature by using Probability Density Function (PDF).

The second type of approaches uses CNN or other deep learning neural networks to learn features directly from the whole signature raw data (e.g., pixels). This kind of approaches has shown their capability of classification [1-3]. For example, Khalajzadeh et al. [12] proposed a CNN-based approach to learning features directly from image pixels of Persian signature datasets and applied their method to writer classification. Soleimani et al. [13] proposed a Deep Multitask Metric Learning (or Deep Multi-Task Metric Learning, DMML) method. They used deep neural networks to learn a distance metric between pairs of signatures.

2.3 Signature Verification by Single Genuine Sample

As we mentioned in Section 2.1, handwriting signatures have high intra-class variability. This has a significant impact on signature verification with single reference signature (because we can't overcome this disadvantage through additional signature samples). Therefore, signature verification with single reference signature has attracted less attention. Adamski and Saeed [4] proposed a method based upon thinning algorithms and sampling techniques to acquire the signature feature vector from one-pixel-wide signature. However, their method only dealt with random forgery samples (by using other users' genuine signatures as forgeries) rather than skilled forgery samples in their experiments. In this paper, our method is to classify one's genuine signature and skilled forgery samples (forgers imitate the genuine signature) which make our research more realistic and reliable.

III. DEEP CONVOLUTIONAL NEURAL NETWORKS

3.1 CNN Architecture

Recent development in deep neural networks has achieved great success in different applications, such as computer vision, natural language processing (NLP), and speech recognition. Convolutional Neural Network (CNN) [14] is a class of deep neural networks. Since Krizhevsky et al. [15] won the ImageNet challenge with CNN in 2012 (AlexNet reached the error rate of 15.3% and was better than the second best entry by 10.4%), CNN has become a most popular and effective method of deep neural networks for computer vision tasks. Some researchers have claimed that they can approach human-level [16] or even super-human performance [17-19].

The CNN architecture used in this research is mainly composed of convolutional layers, pooling layers, and full connected layer as illustrated in Fig. 2. The first two layers run the feature extraction work, and the full connected layer is basically a regular multilayer perceptron (MLP) that connects to the previous layer for the further classification task.

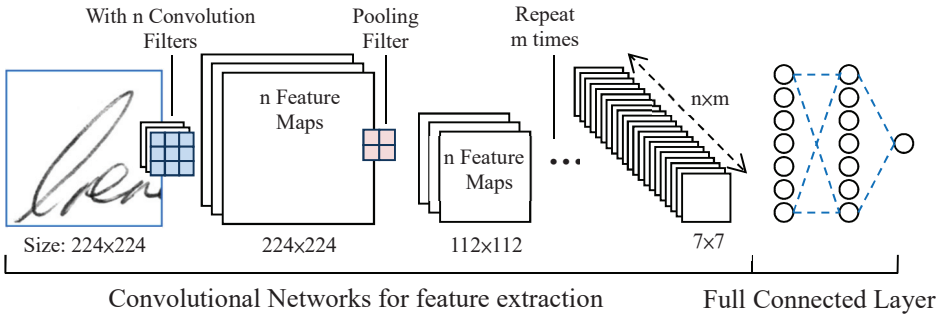


Fig. 2. The CNN architecture used in the proposed method.

3.2 Convolutional Layer

The convolutional layer is used to extract the higher-level features from the low-level information [20], such as detecting the edges, corners, junctions, vertices, end-points and other characteristics from original images. The convolutional layer uses multiple convolution filters or called convolution kernels as a sliding-window to path over the entire image. The sliding area is multiplied by the filters and its sum is saved as a new feature map pixel. Besides, to prevent imperfect overlays on the border, border pixels are computed with zero padding, see Eq. 1 [21].

$$Y(i,j) = (X \times F)(i,j) + b \qquad (1)$$

In Eq. 1, X is the input image, F is the convolution filter, Y is the feature map (the convolved output with an additive bias b). Fig. 3 shows an example of the convolutional operation.

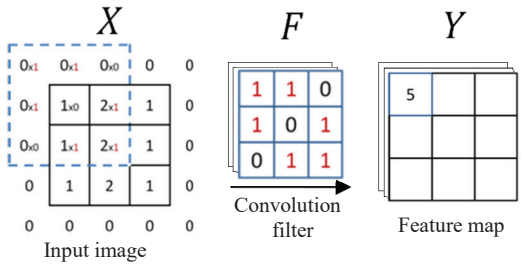


Fig. 3. Convolution filters with zero padding. X is the input images, F is the set of convolution filters, and Y is the set of the feature maps.

In the Deep CNN architecture, the number of hidden layers (mainly constituted by convolution layers) is called the depth of the network. Recent studies [22,23] show that increasing the depth of networks can make the system learn more complex features and is a crucial point of getting good performance. However, the increasing depth of networks will bring dramatic growth in the size of feature maps and accompanied by the tedious computation. Therefore, we use the pooling layer to reduce the feature map size and improve performance.

3.3 Pooling Layer

The pooling layers are used to reduce the feature map size and the spatial dimensions of neural networks. Three are mainly three types of pooling layers: Average Pooling, Min Pooling, and Max-pooling. In this paper, we use Max-pooling for the following two reasons: 1. Max-pooling can theoretically reduce background noises and keep more handwriting features for our signature images. 2. Max-pooling leads to a faster convergence rate for networks [24].

The max-pooling layer used in this paper is a 2x2 sliding window mask moving across the input space (feature maps), and then set the maximum value within this mask as the output (see Fig. 4).

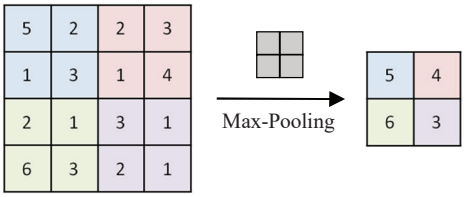


Fig. 4. We use the Max-pooling layer to reduce the feature map size[24,25].

Ⅳ. EXPERIMENTS

In this paper, we use Python 3.6 [26] and Python Imaging Library Pillow 5.0 [27] to implement the data pre-processing. The model training and validation are implemented by using Keras 2.1 [28] and TensorFlow 1.7 [29]. TensorFlow is an open-source deep learning framework released by the Google Brain team in 2015. Keras is an open source high-level neural network library written in Python, and it can run on top of TensorFlow. All our experiments were conducted on a PC with Intel Core i7-4790 Processor (8 logical cores, 8M Cache, up to 4.00 GHz) and 12GB DDR-III RAM.

4.1 Data Acquisition

Our experimental signature samples are from the ICDAR 2011 SigComp dataset [30], which contains different sample sizes of forged signatures for each genuine author, see Table 1.

Table 1. The summary of the forged signature samples in SigComp 2011 datasets(http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2011_Signature_Verification_Competition_(SigComp2011)).

Genuine Samples		Forgeries Samples	
Author ID	Signature Num	Author Num	Signature Num
001	24	2	8
002	24	3	12
003	24	3	12
004	24	3	12
006	24	3	12
009	24	3	12
012	24	3	12
014	**24**	**4**	**16**
015	24	3	12
016	**24**	**4**	**16**

In Table 1, ID No.14 and No.16 contain the largest number of forged reference signatures in the SigComp dataset (16 Skilled Forgeries from 4 authors), so we choose ID No.14 and No.16 author's signatures as experimental samples.

Our experimental dataset includes 1 genuine author with 8 signatures and 4 forged authors with 16 signatures. Note that our genuine author has 24 signature samples and we take the first 8 samples based on the ascending order of signature's ID number in the original SigComp dataset. We denote them as No.1 - No.8 (No.1 as the training data, No. 2-4 as the validation data, and No. 5-8 as the test data, as shown in Table 2).

In this paper, the experiment results and discussions are mainly based on genuine sample No.14. Due to there are some serious defects in the genuine sample No.16 that leads experimental results not reaching the expected accuracy. This will further be discussed in Section 5.4.

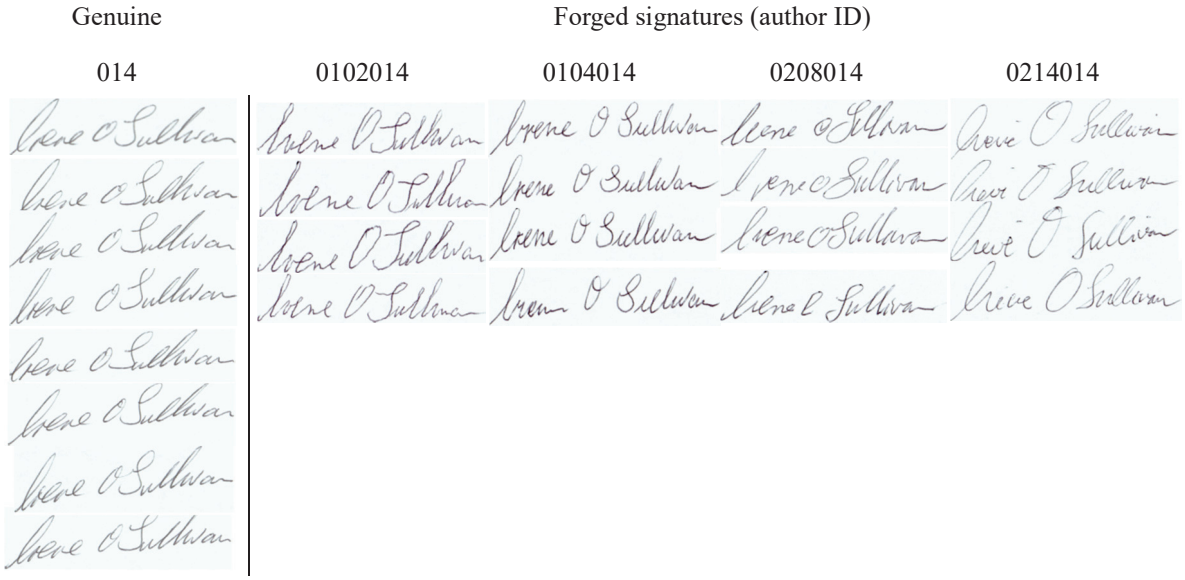


Fig. 5. Experimental data from the ICDAR 2011 SigComp dataset(http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2011_Signature_Verification_Competition_(SigComp2011))

4.2 Data Preprocessing

Since our method only uses a single genuine signature as the training data, we need some steps in the preprocessing for the following purposes. Firstly, our system can increase the number of samples by converting a single original signature image into many sub-image blocks. For example, the first genuine signature author ID 014 is converted to 1062 new images (see Fig. 6), thus we can increase the feasibility of deep learning neural network method. Secondly, using sub-image blocks as the training and identification data can effectively prevent few local features from dominating our CNN system. In some cases, researchers can even fool or attack deep learning systems by inserting designed features, see [30]. It shows that focusing only on a few local features may interfere the verification system.

Thirdly, by applying a rotation process, we can make our CNN system focus on the rotation invariant features and reduce the unnecessary influence from different handwriting angles (rotational invariant). The created rotation data can train our CNN and simulate the author's intra-class variability to a certain extent.

The steps of the preprocessing are shown as follows:

1. we convert the raw images into grayscale images (saved as 24-bit BMP files), as the first step in Fig. 6.
2. We use a block-based method for data expanding (as the second step in Fig. 6). We set a 224×224 window as a sliding mask to get a sub-image block from the original image. Then, we shift the mask by 20 pixels each time to repeat the process from left to right and from top to bottom. After finishing the processes above, we can obtain N overlapping sub-image blocks from the original signature image.
3. We rotate clockwise each sub-image blocks by 60 degrees and repeat the process 5 times to get N×6 sub-image blocks totally (as the third step in Fig. 6). Although we can get more features with smaller rotating angles, we need more computational load.
4. We check each sub-image block to see if it contains enough information for our experiments. To this end, we set some thresholds for inspection based on experience. We set 230 as the threshold grayscale value (the value is set based upon the ICDAR 2011 SigComp dataset). The pixel with the grayscale value over the threshold is regarded as a valid pixel. If a sub-image block has over 7.5% valid pixels, we classify it as a valid sample. Otherwise, we regard it as an invalid sample and drop it.

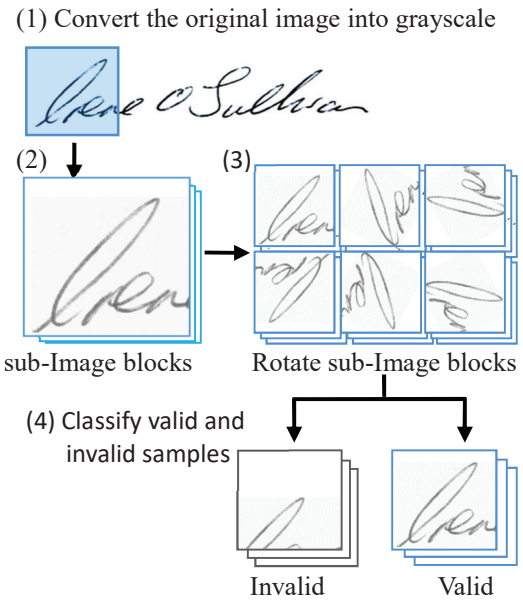


Fig. 6. In the preprocessing, we increase the number of samples by converting a single original signature image into many sub image blocks.

Considering the characteristics of the deep learning network, we deliberately rotate sub-images at a large angle (like 90 degrees and 180 degrees) for better learning efficacy in the step (3). If we limit the rotation angle within -15 to +15 degrees (which is more intuitively close to real writing situation), then the testing accuracy will drop from 98.41% to 94.19%.

4.3 Experimental Design

We designed three experiments with different numbers of training and validation samples, as shown in Table 2. While we use forgery samples from three authors in the first experiment (see Exp.1 in Table 2), we use forgery samples from two authors and one author in the second and third experiments (see Exp. 2 and Exp.3 in Table 2), respectively. We use these three experiments to see if our networks can learn some useful features from signature samples.

Table 2. The summary of the datasets used in three experiments.

author ID		Genuine	Forgery author ID				Number of signatures -> Converted image blocks
		014	0102014	0104014	0208014	0214014	
Exp.1	Training	No.1	No.1,2	No.1,2	No.1,2		(1,6) -> (1602,5914)
	Validation	No.2-4	No.3,4	No.3,4	No.3,4		(3,6) -> (2926,6390)
	Testing	No.5-8				No.1-4	(4,4) -> (4608,2392)
Exp.2	Training	No.1	No.1,2	No.1,2			(1,4) -> (1602,4220)
	Validation	No.2-4	No.3,4	No.3,4			(3,4) -> (2926,4354)
	Testing	No.5-8				No.1-4	(4,4) -> (4608,2392)
Exp.3	Training	No.1	No.1,2				(1,2) -> (1602,1906)
	Validation	No.2-4	No.3,4				(3,2) -> (2926,1944)
	Testing	No.5-8				No.1-4	(4,4) -> (4608,2392)

Note: (x, y) -> (x',y') in this table shows how many sub-image blocks we got from data preprocessing. Where x is the number of genuine signatures and x' is the number of it's sub-images. Likewise y and y' is the number of forged signatures and it's sub-images.

We divide the experimental dataset into three parts for training, validation, and testing (each piece of data only belongs to one part). Take Exp.1 for example, we use genuine signature No.1 of the genuine author 014 and forged signature No.1,2 of three authors (author ID: 0102014, 0104014, 0208014) as training data; genuine signature No.2-4 and forged signature No.3,4 (author ID: 0102014, 0104014, 0208014) as the validation data; genuine signature No.5-8 and forged signature No.1-4 (author ID: 0214014) as the testing data, see Exp.1 in Table 2). Exp.2 and Exp.3 use the data in the same way but with fewer forged signature samples. The dividing of the samples is simply based on the ascending order of author's and signature's ID numbers in the SigComp dataset.

4.4 VGG19 Network Architecture

Simonyan and Zisserman [22] have shown that the depth of the CNN network plays an essential role in classification accuracy. The increasing depth of the network may cause lower performance [32] and over fitting [33]. We use VGG-19 architecture [22] in this paper. There are 27 layers in the VGG-19 architecture: an input layer, 16 convolutional layers, 5 Max-pooling layers, 3 fully-connected layers, Flatten, and output (as shown in Fig. 7). The main layers in our model are of the following types:

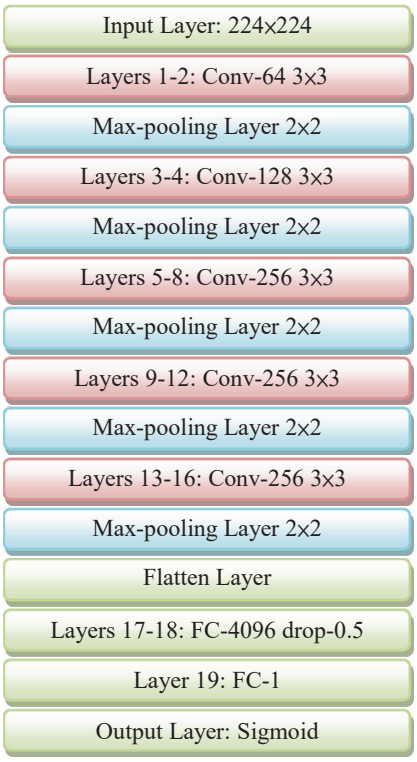


Fig. 7. The CNN architecture used in the experiment (based on VGG19).

1. Conv-64 3×3 represents the convolutional layer with 64 convolution filters in which filter size is 3×3.
2. Max-pooling Layer 2×2 represents the Max-pooling layer with a 2×2 filter.
3. Flatten layer is a utility layer that converts the multidimensional feature map data into a 1D-feature vector.
4. FC-4096 is a classical backpropagation neural network [34]. It represents the fully-connected layer with 4096 perceptrons.

Besides, for the real problems and the real world are mostly nonlinear, we need to add an activation to bring the nonlinear properties into our networks. Since the ReLU function [35] requires less computation (compared to the sigmoid and tanh functions) and can significantly diminish the vanishing gradient problem, we choose ReLU as the activation function in all convolutional and fully-connected layers. ReLU is a non-symmetric function and can be defined as Eq. 2.

$$ReLU(X) = max\{0, X\} \quad (2)$$

In the Layers 17-18, Drop-0.5 means the dropout process [36], which is a widely used technique to prevent Deep Neural Networks from overfitting. During the training phase, we temporarily remove (or ignore) 50% perceptrons by random from the current layer in each training epoch. Finally, we use the sigmoid function in the output layer. The sigmoid function is defined as a curve whose values lie between 0 and 1(see Eq. 3) and can be used for calculating the probabilities in our binary classification task.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

4.5 Training the Networks

Before training our CNN networks, we use a transfer learning [38] technique to save the training time and get better performance. The transfer learning (or inductive transfer) technique is a kind of machine learning method. It takes the model weights from a solved problem which is different but similar to ours. We download the well-trained weights (but not include weights in fully-connected layers) from ImageNet competition for VGG-19 architecture [39] as our initial model weights. This network is built to recognize 1000 different categories (such as cat, bike, airplane, etc.) from target Images. It is already finely tuned and does exceptionally well in object detection and classification. We use those weights as pre-trained weights to provide a shortcut to train our networks more efficiently.

In the training process, we need a loss function to estimate the deviance between the predicted and actual labels. The lower the loss value, the closer network predictions are to the correct labels. Considering that our output layer is a sigmoid function, we use a cost function called Binary Cross-Entropy (BCE) as shown in Eq. 4, where $\hat{y}$ is the predicted probability in which target is a genuine signature and y is the correct one [39].

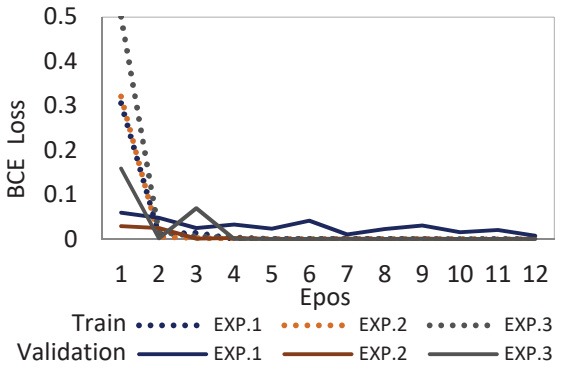
$$BCE = -y\,log(\hat{y}) - (1 - y)\log(1 - \hat{y}) \quad (4)$$

We use the Stochastic Gradient Descent (SGD) algorithm [40] to update all parameters in our network. SGD is a popular optimization technique for the deep learning problem. Since an extortionate learning rate may hinder the convergence, we set a relatively small value of 1e-4. The momentum is set as 0.9, which is the most often used value in SGD.

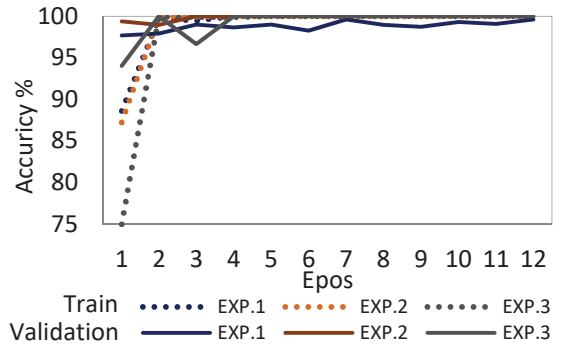
V. RESULTS AND DISCUSSION

5.1 Networks Training and Validation

Our dataset includes 24 images (with 1 genuine and 4 forged signatures). All images are converted into 23,832 image blocks for the experiment. Our training is finished in 12 epochs, due to the training and validation performance does not significantly improve and tend to be stable (each epoch represents a full pass over the training or validation data). From Exp.1 to Exp.3, we get nearly 100% training accuracy; 99.6%, 99.99%, and 100% validation accuracy; 7.44E-05, 6.48E-05, and 1.12E-04 training loss; 0.0077, 9.48E-04, and 2.21E-04 validation loss, as illustrated in Figs. 8a and 8b.



(a)



(b)

Fig. 8. (a) BCE Loss per epoch during the training and validation, (b) accuracy per epoch during the training and validation

5.2 Networks Performance

After the model training and validation, we test the network with a new author (ID: 0214014) which is not in the training and verification dataset (see Table 2). Then we evaluate our performance by using the following metrics: Accuracy, False Rejection Rate (FRR) or Type I error(α), False Acceptance Rate (FAR) or Type II error(β), which are defined in Eqs. 5-7 [41].

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \times 100\% \quad (5)$$

$$\text{FRR} = \alpha = \frac{\text{FN}}{\text{TP} + \text{FN}} \times 100\% \quad (6)$$

$$\text{FAR} = \beta = \frac{\text{FP}}{\text{FP} + \text{TN}} \times 100\% \quad (7)$$

TP means the number of true positive (genuine signatures are correctly classified as positive). Likewise, we got false positive (FP), false negative (FN), and true negative (TN). Thus, the higher accuracy and lower error rate (FRR / FAR) can be considered the better performance. In this research, different results obtained considering different sample size (numbers of forgery authors and signatures for training) are summarized in Table 3.

Table 3. The summary of experimental results.

Category	Exp.1	Exp.2	Exp.3
Forged authors for training	3	2	1
Forged signatures for training	6	4	2
Train accuracy (%)	100	99.83	99.97
Validation accuracy (%)	99.60	99.9	100
Test accuracy (%)	98.41	90.31	80.1
Test FAR (%)	0.91	0.54	0
Test FRR (%)	2.88	27.3	58.24

Note: Each forgery author has four signature samples in the original dataset

5.3 Discussion

The results of our main task (Exp.1) are very encouraging. Our approach not only correctly distinguishes the genuine and forgeries in the training and validation period but also achieves the accuracy of 98.41%, FAR of 0.91% and FRR of 2.88% in testing dataset. Please be noted that each signature sample is only used in one of the training, validation and testing sets. Furthermore, the signatures used in the testing process are collected from a whole new author whose samples are not present in the training and validation processes. It indicates that our approach is able to learn some useful features to discriminate among different signatures and authors. Experiment Exp. 2 and Exp. 3 are also very useful because we can observe that the accuracy drops significantly with the decreasing number of forgery samples. It shows the factor that there is no sufficient information can learn in those experiments with fewer signature samples.

While we have good experimental results with the genuine sample No.14, we obtain quite different results with the genuine sample No.16, as we mentioned in Section 4.1. We repeat the experiments for the genuine sample No.16 and get the accuracy of 60.25%, FAR of 36.95 % and FRR of 42.21% in the testing dataset. We check

the samples of No.16 and find the samples are with high intra-class variability. The genuine signatures used for training and testing processes are shown in Fig. 9. Obviously, it is very difficult to verify if these signatures are from the same person, even with human visual comparison. Since our signature verification method is only based upon single genuine sample, it may lead to poor performance (low accuracy, FAR, and FRR) when training and testing data have high intra-class variability. Therefore, the experimental results make sense.

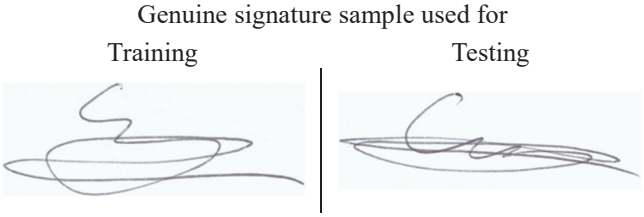


Fig. 9. High variability between samples (written by author No.16.)

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose an off-line handwritten signature verification method by using a unique local feature extraction approach and CNN. Our CNN is trained by only single genuine and several forged signature samples in the training process. From the experimental results, the proposed method can achieve an accuracy of 98.41% with very few training samples (single genuine signature and six forgeries in Exp.1), and even under the most extreme conditions (single genuine signature and only two forgeries in Exp.3), our method still maintains accuracy of 80.1%. In practical applications, since more forgery samples will give more information, we can easily obtain or even create forged signatures by ourselves to improve the performance of the proposed method.

Our method is currently for solving a binary classification problem. We use the sigmoid function as the output layer. The softmax layer outputs a value between 0 and 1 to represent the probability of one class (genuine signature's sub-image in this case), so the probability of the other class is just 1-p. If we want to verify the final result, we may use the voting method based on the number of sub-images. However, if we need the optimal verification results, more data and follow-up studies are needed.

This paper shows the capability of the CNN model on the signature examination. Our experiments use the open databases that do not consider the effects of many handwriting variables, such as the thickness, paper types, various forgery skills, etc. Our further work will focus on these subjects.

REFERENCES

[1] Alvarez G., Sheffer B., and Bryant M., "Offline Signature Verification with Convolutional Neural Networks," Tech. rep. Stanford University, Stanford, 2016.

[2] Zhang X. Y., Bengio Y., and Liu C. L., "Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark," Pattern Recognition, Vol. 61, pp. 348-360, 2017.

[3] Hafemann L. G., Sabourin R., and Oliveira L. S., "Offline handwritten signature verification—literature review," Image Processing Theory, Tools and Applications (IPTA), 2017 Seventh International Conference, pp. 1-8, 2017.

[4] Adamski M. , and Saeed K., "Signature verification by only single genuine sample in offline and online systems," AIP Conference Proceedings, Vol. 1738, No. 1, pp. 180011, 2016.

[5] Justino E. J. R., ElYacoubi A., Bortolozzi F., and Sabourin R., "An off-line signature verification system using HMM and graphometric features," Proc. of the 4th international workshop on document analysis systems, pp. 211-222, 2000.

[6] Leclerc F., and Plamondon R., "Automatic signature verification: The state of the art—1989–1993," International journal of pattern recognition and artificial intelligence, Vol. 8, No. 03, pp. 643-660, 1994.

[7] Plamondon R., and Srihari S. N., "Online and off-line handwriting recognition: a comprehensive survey," IEEE Transactions on pattern analysis and machine intelligence, Vol. 22, No. 1, pp. 63-84, 2000.

[8] Impedovo D., Pirlo G., and Plamondon R., "Handwritten signature verification: New advancements and open issues," [w:] Frontiers in handwriting recognition (ICFHR), 2012 international conference on, pp. 367-372, 2012.
[9] Shah A. S., Khan M. N. A., and Shah A., "An appraisal of off-line signature verification techniques," International Journal of Modern Education and Computer Science, Vol. 7, No. 4, pp. 67, 2015.
[10] Deng P. S., Liao H. Y. M., HoC. W., and Tyan H. R., "Wavelet-based off-line handwritten signature verification," Computer Vision and Image Understanding, Vol. 76, No. 3, pp. 173-190, 1999.
[11] Drouhard J. P., Sabourin R., and Godbout M., "A neural network approach to off-line signature verification using directional PDF," Pattern Recognition, Vol. 29, No. 3, pp. 415-424, 1996.
[12] Khalajzadeh H., Mansouri M., and Teshnehlab M., "Persian signature verification using convolutional neural networks," International Journal of Engineering Research and Technology, Vol. 1, pp. 7-12, 2012.
[13] Soleimani A., Araabi B. N., and Fouladi K., "Deep multitask metric learning for offline signature verification," Pattern Recognition Letters, Vol. 80, pp. 84-90, 2016.
[14] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., HubbardW., and Jackel L. D., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, Vol. 1, No. 4, pp. 541-551, 1989.
[15] Krizhevsky A., Sutskever I., and Hinton G. E., "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, pp. 1097-1105, 2012.
[16] Taigman Y., Yang M., Ranzato M., and Wolf L., "Deepface: Closing the gap to human-level performance in face verification," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1701-1708, 2014.
[17] Zhong Z., Jin L., and Xie Z., "High performance offline handwritten chinese character recognition using googlenet and directional feature maps," Document Analysis and Recognition (ICDAR), 2015 13th International Conference, pp. 846-850, 2015.
[18] He K., Zhang X., Ren S., and Sun J., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," Proceedings of the IEEE international conference on computer vision, pp. 1026-1034, 2015.
[19] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., and Bernstein M., "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, Vol. 115, No. 3, pp. 211-252, 2015.
[20] Girshick R., Donahue J., Darrell T., and Malik J., "Rich feature hierarchies for accurate object detection and semantic segmentation," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587, 2014.
[21] Goodfellow, I., Bengio, Y., and Courville, A., Deep Learning, MIT Press, Chap. 9, pp. 333, 2016.
[22] Simonyan K., and Zisserman A., "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[23] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., and Rabinovich A., "Going deeper with convolutions,"Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9, 2015.
[24] Scherer D., Müller A., and Behnke S., "Evaluation of pooling operations in convolutional architectures for object recognition," Artificial Neural Networks–ICANN, Springer, pp. 92-101, 2010.
[25] Jarrett K., Kavukcuoglu K., and LeCun Y., "What is the best multi-stage architecture for object recognition," Computer Vision, IEEE 12th International Conference on, pp. 2146-2153, 2009.
[26] Python Foundation, "About PythonTM | Python.org," About Python, https://www.python.org/.
[27] Lundh F., and Clark A., "Pillow: the friendly PIL fork," https://python-pillow.org/.
[28] Abadi M., Agarwal A., Barham P., Brevdo

E., Chen Z., Citro C., Corrado G. S., Davis A., Dean J., and Devin M., "TensorFlow: large-scale machine learning on heterogeneous systems," URL https://www. tensorflow. org.
[29] Chollet F., "Keras Documentation," https://keras.io/.
[30] Liwicki M., Malik M. I., Van DenHeuvel C. E., Chen X., Berger C., Stoel R., Blumenstein M., and Found B., "Signature verification competition for online and offline skilled forgeries (sigcomp2011)," Document Analysis and Recognition (ICDAR), 2011 International Conference on, pp. 1480-1484, 2011.
[31] Brown T. B., Mané D., Roy A., Abadi M., and Gilmer J., "Adversarial patch," arXiv preprint arXiv:1712.09665, 2017.
[32] He K., and Sun J., "Convolutional neural networks at constrained time cost," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5353-5360, 2015.
[33] Gu J., Wang Z., Kuen J., Ma L., Shahroudy A., Shuai B., Liu T., Wang X., Wang L., and Wang G., "Recent advances in convolutional neural networks," arXiv preprint arXiv:1512.07108, 2015.
[34] Bishop C., and Bishop C. M., "Neural networks for pattern recognition,". Oxford university press, 1995.
[35] Nair V., and Hinton G. E., "Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807-814, 2010.
[36] Hinton G., and Dahl G., "Dropout: A simple and effective way to improve neural networks," Advances in Neural Information Processing Systems, 2012.
[37] Pan S. J., and Yang Q., "A survey on transfer learning," IEEE Transactions on knowledge and data engineering, Vol. 22, No. 10, pp. 1345-1359, 2010.
[38] Chollet F., "Keras," https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5.
[39] DeBoer P.-T., Kroese D. P., Mannor S., and Rubinstein R. Y., "A tutorial on the cross-entropy method," Annals of operations research, Vol. 134, No. 1, pp. 19-67, 2005.
[40] Bottou L., "Large-scale machine learning with stochastic gradient descent," Proceedings of COMPSTAT'2010, Springer, pp. 177-186, 2010.
[41] Scharcanski J., Proença H., and Du E., Signal and image processing for biometrics, Springer Press, Chap. 5, pp. 121, 2014.