

Article

# Forged Signature Distinction Using Convolutional Neural Network for Feature Extraction

Seungsoo Nam <sup>1</sup>, Hosung Park <sup>2</sup>, Changho Seo <sup>1</sup> and Daeseon Choi <sup>2,\*</sup>

<sup>1</sup> Department of Convergence Science, Kongju National University, Chungnam 32588, Korea; tfok815@kongju.ac.kr (S.N.); chseo@kongju.ac.kr (C.S.)

<sup>2</sup> Department of Medical Information, Kongju National University, Chungnam 32588, Korea; hspark@kongju.ac.kr

\* Correspondence: sunchoi@kongju.ac.kr; Tel.: +82-41-850-0345

Received: 8 December 2017; Accepted: 19 January 2018; Published: 23 January 2018

**Abstract:** This paper proposes a dynamic verification scheme for finger-drawn signatures in smartphones. As a dynamic feature, the movement of a smartphone is recorded with accelerometer sensors in the smartphone, in addition to the moving coordinates of the signature. To extract high-level longitudinal and topological features, the proposed scheme uses a convolution neural network (CNN) for feature extraction, and not as a conventional classifier. We assume that a CNN trained with forged signatures can extract effective features (called *S-vector*), which are common in forging activities such as hesitation and delay before drawing the complicated part. The proposed scheme also exploits an autoencoder (AE) as a classifier, and the *S-vector* is used as the input vector to the AE. An AE has high accuracy for the one-class distinction problem such as signature verification, and is also greatly dependent on the accuracy of input data. *S-vector* is valuable as the input of AE, and, consequently, could lead to improved verification accuracy especially for distinguishing forged signatures. Compared to the previous work, i.e., the MLP-based finger-drawn signature verification scheme, the proposed scheme decreases the equal error rate by 13.7%, specifically, from 18.1% to 4.4%, for discriminating forged signatures.

**Keywords:** dynamic signature; convolution neural network; autoencoder neural network; skilled forgery; biometric

## 1. Introduction

With increasing smartphone usage, several studies have focused on user authentication based on behavior characteristics that can be acquired on a smartphone; these include keystroke dynamics [1], gaze tracking [2], and dynamic signature [3]. Among these studies, dynamic signature verification is an authentication method that employs both the uniqueness of the signature and behavioral characteristics corresponding to signing. Similar to other biometrics, a dynamic signature is safe from loss and theft. In addition, the user is less resistant to using signatures for authentication than other biometrics such as fingerprints and iris.

Users can produce a signature on the smartphone screen by using a stylus or their finger. Early studies [4–10] primarily focused on verifying signatures drawn by a stylus. Finger-drawn signature verification [3,11–13] is relatively not well researched yet; however, it is currently actively discussed because most smartphones do not have a stylus, and finger signing is more convenient. The verification of finger-drawn signatures involves new challenges because it results in less precise signals than signatures drawn using a stylus. Moreover, many effective features such as pen pressure, pen angle, and the number of times a pen is lifted, cannot be employed.

To improve the accuracy of signature verification, some studies [8–10] utilize machine learning techniques, which are one of the most noteworthy technologies. Buriro et al. [3] used multilayer

perceptron (MLP), a two-class classifier, for finger-drawn signature verification with dynamic features involving finger and phone movements. Their method exhibited a verification accuracy of 94.8% for classifying the subject and other objects. However, this technique does not represent the capability of distinguishing forged signatures. Two-class classifiers such as MLP are at greater risk of misclassifying a forged signature as the subject, because forged signatures closely resemble the subject [10,14]. Although a signature marked on a smartphone screen disappears immediately after verification, it is easy for an adversary to imitate a signature through shoulder surfing and smudging, which involves tracing the smudge remaining on the screen [11]. Therefore, the verification for distinguishing forged signatures must be improved to provide secure services. Further, the difference in the time between registering a signature and verifying a new signature is also an important issue. Unlike biometrics, such as those involving the fingerprint and iris, behavioral patterns can change with time. Although [15] focused on the issue of this time difference, they worked towards verifying a PIN number, and not dynamic signatures, drawn using the finger.

In this paper, we propose a novel dynamic verification approach for finger-drawn signatures, which provides improved accuracy against forged signatures and time-deferred signatures. The proposed method exploits two deep learning algorithms: Convolutional neural network (CNN) [16,17] for feature extraction, and an autoencoder (AE) [18] as a classifier. CNNs are trained for distinguishing forged signatures from genuine signatures, and the trained CNN is used for feature extraction and not as a classifier. Specifically, the output of the CNN's intermediate layer, which we call *S-vector*, is used as the input to an AE for building the subject model. Because a CNN is known to be capable of extracting features for classification by itself [17], we assume that the CNN trained for a specific purpose could extract features effective for that purpose. For example, a CNN trained with forged signatures can extract features that are common in forging activities such as hesitation and delay before drawing the complicated part of a signature. The proposed method exploits AE as a classifier owing to its high accuracy in solving one-class distinction problems like user authentication. Our previous work [19] and M. Fayyza et al. [10] showed one-class model AE is better at distinguishing imitated signatures than two-class model. However, AE is also greatly dependent on the accuracy of input data [14]. The *S-vector* is valuable as the input of AE, and consequentially could lead to improved accuracy for dynamic signature verification. Even though the major issue is distinguishing forged signature, the proposed method should take into account other issues, i.e., time difference issue and subject/others classification. They are also important issues for the signature verification service and a model for only a single purpose could degrade other issues' performances. The proposed method obtains the better performance for time-differed signatures and subject/others classification by experimental approach.

The main contributions of this paper are as follows.

- To the best of our knowledge, the proposed scheme is the first CNN-AE model for hand drawn signature verification in mobile environments.
- Experiments using users' real signatures show that the *S-vector* achieves better accuracy for dynamic signature verification. The proposed scheme decreases the equal error rate (EER) by 1.1% (subject/others), 3.2% (time difference), and 13.7% (skilled forgery) compared to the previous work.

This paper is organized as follows. Section 2 introduced the proposed method is introduced. In Section 3, the proposed model is evaluated based on experimental results, wherein the CNN was trained with four different classification datasets to determine the most effective *S-vector*. Section 4 concludes the paper.

## 2. Related Work

Signature verification techniques can be classified into two approaches: function-based and feature-based [20]. Function-based signature verification refers to the matching process using the

original time-series data of a signature. The feature extraction process could be absent. Well-known function-based approaches include the dynamic time warping (DTW) algorithm [21], root mean square method (RMS) [22], and hidden Markov model (HMM) [23,24]. Generally, function-based systems demonstrate better verification performance than feature-based systems [24]. However, during the matching process, a dynamic construction of the original signature is revealed, resulting in a potential privacy breach if the matching must be done remotely [11]. Furthermore, the system is more complex and requires longer execution time than feature-based systems [4].

Feature-based signature verification refers to the process of statistical similarity comparison using descriptive features of a signature. A signature is accepted as being genuine if the similarity score exceeds a certain threshold. Feature-based systems are lightweight, less complex, and faster than function-based systems [4]; thus, they are suitable for mobile environments. The main issues of feature-based systems pertain to the extraction of effective features and application of efficient verification algorithms. F. Aguilar et al. [5] proposed a set of diverse features such as total duration of the signature, pen pressure, and the number of times the pen is lifted.

Based on this feature set, Nanni [6] proposed a multimatcher method to verify signatures. In addition, Guru and Prakash [7] introduced a symbolic representation of an online signature.

To improve the accuracy of signature verification, some studies [8–10] utilize machine learning techniques, which are one of the most noteworthy technologies. Similar to the process in feature-based signature verification, they use descriptive features of signatures for building a subject model. Iranmanesh et al. [9] exploited MLP for signature verification, and demonstrated an average accuracy of 82.42% for recognizing the subject; however, this method could not distinguish between imitated and original signatures. As mentioned previously, M. Fayyza et al. [10] used AE, the one-class model, to distinguish imitated signatures. They derived the  $(x, y)$  coordinates, sign changes  $(dx/dt)$ ,  $(dy/dt)$ , pen pressure, and pen angle as features. The verification accuracy was 92% for classifying subject and imitated signatures.

However, all of the above-mentioned feature-based signature verification schemes focus on verifying signatures drawn by stylus instead of finger drawn signature. As mentioned above, verification of finger drawn signatures currently receives great attention but is still a challenging issue due to the lack of precise signals and features. N. Sae-Bae et al. [11], M. Antal et al. [12], and N. Paudel et al. [13] propose their own matching algorithms, not the machine learning techniques, for finger drawn signature verification. A. Buriro [3] applies deep learning algorithm for finger drawn signature verification using dynamic features such as finger movements and phone movements. However, they exploit MLP, two-class classifier similar to Iranmanesh et al. [9], and suffer from the same problem, i.e., trouble of distinguishing imitated signatures.

This work focuses on the finger-drawn signature verification scheme using deep learning. To efficiently distinguish the imitated signatures, the proposed method exploits AE as the one-class model. As mentioned above, finger-drawn signature verification has limitation owing to less precise signals and fewer features. The proposed method therefore uses CNN for feature extraction to improve the verification accuracy.

### 3. Proposed Method

With the purpose of this study, the design principles of the proposed method are as follows.

- Provide user authentication result based on the hand-drawn signature.
- Provide the better accuracy for distinguishing skilled forgery.
- Reduce the accuracy decline caused by time difference.

#### 3.1. Data Acquisition of Dynamic Signatures on a Smartphone

Signatures on mobile devices are often drawn by a user's finger instead of a stylus and an example of finger-drawn signatures is shown in Figure 1. When a user signs on a smartphone screen, the coordinates of points  $(Px, Py)$  and the sensed value of the accelerometer  $(Ax, Ay, Az)$  are

sampled every 32 ms. One such sample  $x_i$  is defined in Equation (1), where  $dis_i$  is the distance from  $(Px_{i-1}, Py_{i-1})$  to  $(Px_i, Py_i)$ . The entire signature is denoted as  $X$ , which is a sequence of  $x_i, i = 0, 1, \dots, n - 1$  for the  $n$  samples comprising the signature.

$$x_i = (Px_i, Py_i, Ax_i, Ay_i, Az_i, dis_i), \tag{1}$$

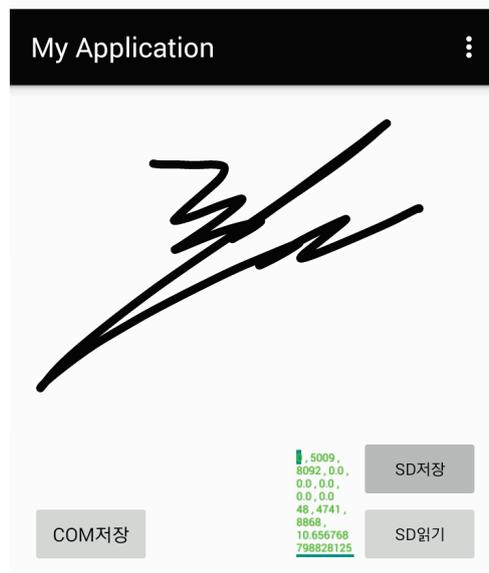


Figure 1. Original signature.

### 3.2. Data Normalization

The acquired data from all signatures is pre-processed by normalizing the alignment, size, and length before feature extraction. Because signatures have different values of starting point, size, and the number of samples, the data must be adjusted to equivalent conditions for more precise learning, modeling, and verification. To standardize starting points, each signature is aligned so that the starting point is set to  $(0, 0, 0, 0, 0)$ . The processed data by alignment normalization  $Xa$  is calculated using Equation (2), where  $(Px_0, Py_0, Ax_0, Ay_0, Az_0)$  denotes the original starting point of the signature. The distance  $dist_i$  is omitted because it is unrelated to the starting points.

$$Xa = (Px_i - Px_0, Py_i - Py_0, Ax_i - Ax_0, Ay_i - Ay_0, Az_i - Az_0), \quad i = 0, \dots, n \tag{2}$$

Even though the signatures are drawn on a restricted plane, i.e., smartphone screen, each signature is different in size. This size variation could lead to performance degradation of signature verification. Therefore, the proposed scheme makes the signatures into similar size by size normalization which ensures that all values of points range between 0 and 1. The processed data by size normalization  $Xs$  is calculated from  $Xa$  using Equation (3), where  $\max(Px)$  is the maximum value among  $Px_0, Px_1, \dots, Px_n$ .

$$Xs = \left( \left( \frac{Px_i - \max(Px)}{\min(Px) - \max(Px)}, \frac{Py_i - \max(Py)}{\min(Py) - \max(Py)}, \frac{Ax_i - \max(Ax)}{\min(Ax) - \max(Ax)}, \right. \right. \tag{3}$$

$$\left. \left. \frac{Ay_i - \max(Ay)}{\min(Ay) - \max(Ay)}, \frac{Az_i - \max(Az)}{\min(Az) - \max(Az)} \right) \right), \quad i = 0, \dots, n$$

Length normalization refers to the process of equalizing signature length, i.e., the number of samples. Each signature length is normalized to a mean length of the subject from  $Xs$  using Equations (4) and (5), where  $n$  is the original length of  $Xs$ ,  $m$  is the mean length of the subject,  $k$  is rate for equal

division, and  $t$  is a proper integer.  $x'_0$  and  $x'_{m-1}$  are, respectively, the first and last samples. They are kept constant to equalize the starting and ending points of all signatures. After length normalization, the sequence of  $x'_i$ , i.e.,  $X'$ , is a signature normalized using three normalization processes.

$$\begin{aligned}
 k_0 &= K_1 = n/m \\
 K_{j+2} &= k_{j+1} + k_j, \quad j = 0, \dots, m - 3
 \end{aligned}
 \tag{4}$$

$$\begin{aligned}
 &i = 1 \mid i \leq m - 2 \\
 &\text{if } t \leq k_i < t + 1 \\
 &x'_i = x_t + (x_{t+1} - x_t) \times (k_i - t) \\
 &x'_0 = x_0 \\
 &x'_{m-1} = x_{n-1}
 \end{aligned}
 \tag{5}$$

Figure 2 shows an example of data normalization. Figure 2a shows  $Xa$ , with the starting point of  $(0, 0)$ . Figure 2b shows  $Xs$ , with all values existing between 0 and 1. Figure 2c shows  $X'$ . In addition, the number of points, i.e., the number of samples, are changed to the mean length of the subject.

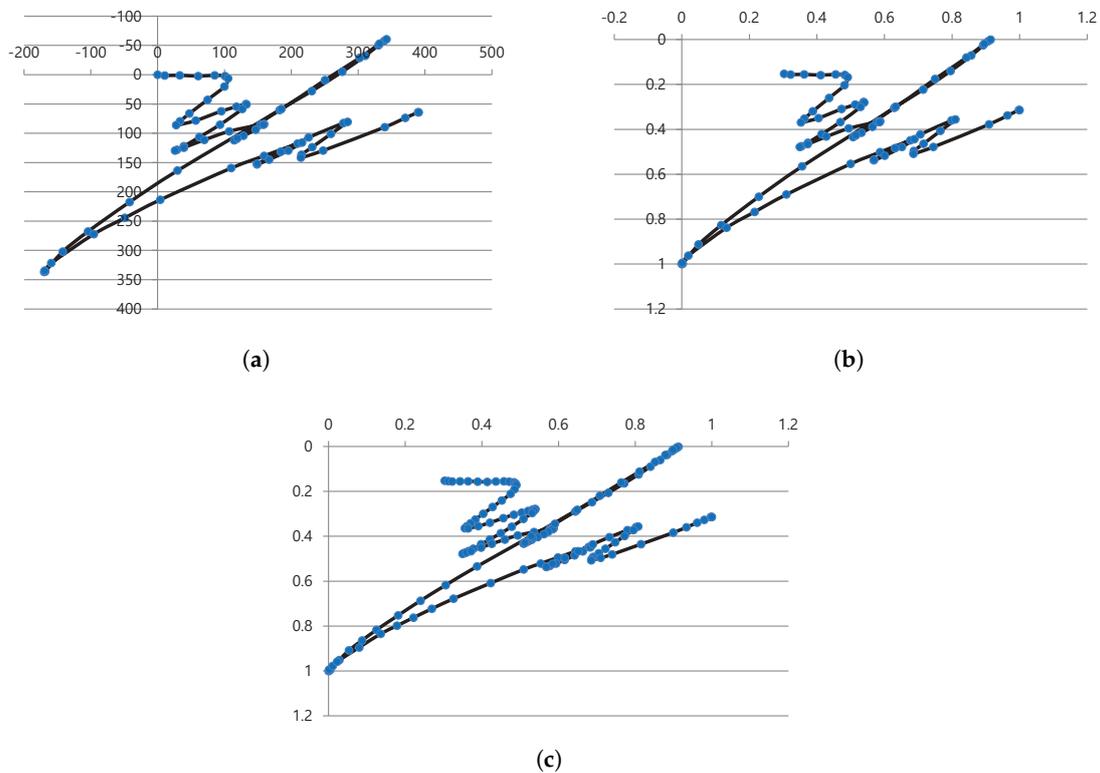


Figure 2. Data normalizations of: (a) alignment; (b) size; and (c) length.

Figure 3 shows the sequence of  $x'_i$ , i.e., a normalized signature  $X'$  according to the sequence of 150 samples.

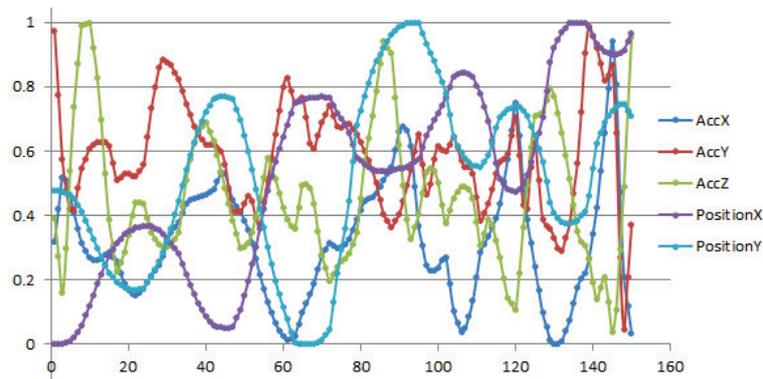


Figure 3. A normalized signature  $X'$  (sequence of  $x'_i$ ).

### 3.3. CNN for Feature Extraction

A convolutional neural network (CNN) is a multi-layer neural network with a deep supervised learning architecture that is known to have the capability of extracting features for classification by itself [17]. A CNN is composed of two parts: an automatic feature extractor and a trainable classifier. The feature extractor extracts features from input data via two operations: convolution filtering and down sampling. Based on these features, the trainable classifier is trained using a back-propagation algorithm with a fully connected layer, and it produces the classification results.

The proposed method uses a CNN only as a feature extractor, not as a classifier. Figure 4 shows the architecture of the proposed CNN-AE model. Similar to other deep neural networks, in CNN, the extraction process is black box and the exact characteristics of the features cannot be known. We assume that if a CNN is trained for classifying forged and genuine signatures, the trained CNN can extract effective features for distinguishing behavior characteristics of forgery, such as hesitation and delay before drawing the complicated part of a signature. Therefore, the output of the CNN feature extractor is used as a feature vector, defined as the *S-vector* (denoted by  $S$  in figures and equations). The *S-vector* is used as the input to an AE for building the subject model. As mentioned earlier, an AE has high accuracy for the one-class distinction problem such as user authentication. The AE operation is represented in the next subsection.

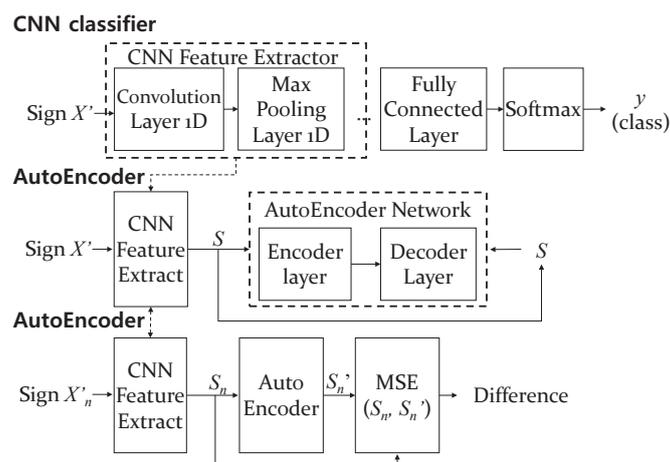


Figure 4. Architecture of the proposed CNN-AE model.

In the CNN feature extractor, the convolution layer uses  $X'$  as the input value. Each node  $c_i$  of the convolution layer is defined in Equation (6), where  $k$  is a kernel (also called filter) and  $l$  is the number of kernels used.

$$c_i = \sum_{j=0}^l k_j \times x'_{i+j} \tag{6}$$

The output of  $c_i$  is input to an activation function  $ReLU$  defined in Equation (7). The activation results constitute a convolution map. These procedures of the convolution layer are shown in Figure 5.

$$ReLU(c_i) = \max(0, c_i) \tag{7}$$

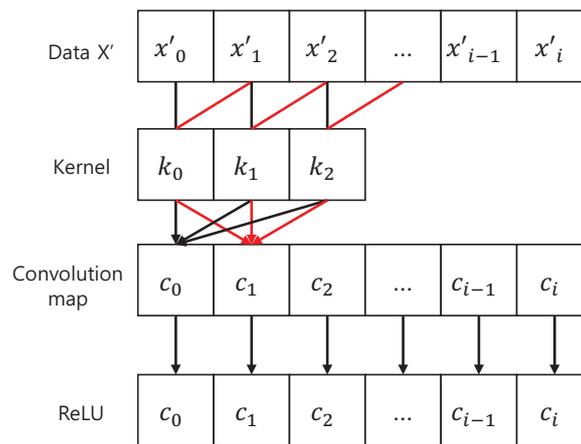


Figure 5. Operation of the one-dimensional convolution layer.

The convolution map is filtered for allowing some variance in the data input by the pooling layer. Max pooling [25] is used and one pooling node is defined in Equation (8), where the value 2 is used as a stride (moving  $n$  pixels in a layer) that controls the pooling range.

$$F_i = \max(c, c_{i+1}), \quad i = 2, 4, \dots, n \tag{8}$$

Convolution and pooling can be repeated multiple times. We do not discuss the remaining layers and training method here, as they can be referred to in [17]. The outputs of the last pooling layer constitute the  $S$ -vector denoted by  $S$  that is a concatenation of  $F_i$  as defined in Equation (9).

$$S = F_0|F_1|\dots|F_k, \quad k = 0, \dots, n \tag{9}$$

The training data set may vary according to the purpose of the CNN classifier. For example, the CNN can be trained with the genuine and forged signatures as two-class input data. In this case, the  $S$ -vector can be effective for distinguishing forged signatures, although it can have greater risk of misclassifying subject/others. In contrast, in the case of a CNN trained using the subject's and others' signatures, the  $S$ -vector can be effective for classifying subject/others but can have greater risk of misclassifying subject/forged. Furthermore, models for each subject and a model for all subjects yield different results. We experiment with four different class types described in Section 4.1.

### 3.4. AE (Autoencoder) for Subject Modeling

The  $S$ -vector extracted by the CNN feature extractor is used as the input of an AE. An AE is a type of deep neural network that has the same dimensions for input and output [14]. In the training phase, the AE is trained by using the same data (sample signatures) as input and output. In the test phase, the trained AE generates an output corresponding to an input (test signature). An AE can generate

highly similar output for trained data patterns, whereas it does not for unfamiliar data. Therefore, a test signature is verified by a similarity comparison between the test signature and the output of the AE. Consequently, an AE is used in modeling a subject for authentication.

The proposed scheme uses a five-layer AE, as illustrated in Figure 6. Note that the entire architecture of the CNN-AE model is depicted in Figure 4. The AE itself consists of two blocks: encoder and decoder. The outputs of the encoder and decoder are denoted as  $z$  and  $S'$ , respectively, and defined as Equations (10) and (11), respectively. The encoder accepts  $S$  as the input and produces  $z$  that is implemented as hidden nodes in the neural network. The decoder accepts  $z$  and produces  $S'$ .  $\sigma_1$  and  $\sigma_2$  are the activation functions.

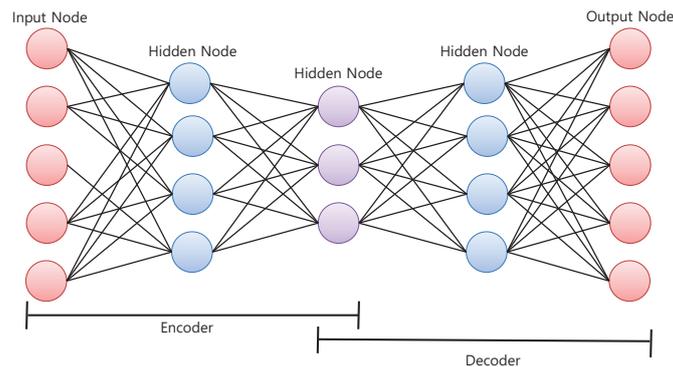


Figure 6. Structure of autoencoder.

$$z = \sigma_1(WS + b) \tag{10}$$

$$S' = \sigma_2(W'z + b') \tag{11}$$

The loss function that is the objective for training a neural network is defined as in Equation (12). It is the sum of all differences between the input and output.

$$L(S, S') = \sum |S - S'| \tag{12}$$

The training of the AE is a procedure involving the modification of weights  $W$  and  $W'$  in Equations (10) and (11), respectively, for minimizing  $L$  with  $S$  as both input and output. In the test phase, which is the verification procedure in this problem, a newly input signature  $S$  to be verified is entered into the AE and compared with the output  $S'$  of the AE. The proposed scheme applies mean square error (MSE) for the similarity comparison. MSE refers to the difference between  $S$  and  $S'$  and is defined as Equation (13). Using MSE, the verification rule is defined as Equation (14). If the difference is less than a pre-defined threshold  $t$ , the signature is accepted as a valid one.

$$Diff = MSE(S, S') \tag{13}$$

$$Diff < t, t : \text{threshold} \tag{14}$$

## 4. Experiments

### 4.1. Experimental Settings

As no public test set of dynamic signatures has been conducted on smartphones, we constructed a test set by ourselves. We gathered 20 signatures per person from 20 subjects, that we denoted as the original signature. After a month, we acquired 20 signatures per person from 10 subjects, that we denoted as the time-deferred signatures. Each subject's signature was shown to four forgers for imitating five signatures. Thus, 20 forged signatures were created for each subject. Figure 7 shows

the samples of original signatures and forged signatures. Experiment participants were composed of 16 people in their twenties, 3 in their thirties, and 1 in their forties. The gender ratio is 15 males and 5 females. They signed their real signatures with a finger, holding a smartphone with other hand. The smartphone device was Galaxy S3 and the embedded application for signature registration was used. Average signing time was about 3 min for 20 signatures.



Figure 7. Samples of: (a) original signatures; and (b) forged signatures.

The experiments have some limitations. For the accurate and practical experiment, we collected the real signatures of the users. The data set therefore is relatively small due to a privacy issue. Signatures are also drawn on same model of smartphone. If the type of smartphone is different, some technical details such as sampling rate may also be different. Therefore, other normalization process may be needed. The accuracy decline caused by time difference is an inherent problem in behavioral biometrics authentications. The proposed method obtains the better accuracy for time-deferred signatures by experimental approach but could not suggest explicit solution.

Using different configurations of the training data, four different CNN feature extractors were created, as shown in Table 1. CNN\_A is a two-class classifier trained with all the subjects' original signatures as one class and all the subjects' forged signatures as the other class. CNN\_B is a 20-class classifier trained with 20 subjects' original signatures. CNN\_C is made for each subject with the subject's and others' original signatures. CNN\_D is made for each subject with the subject's original and forged signatures. CNN\_D is possible only in an experimental setup as it is impossible to acquire forged signatures for a newly enrolled subject. However, we included CNN\_D for evaluating the

effect of a CNN trained for a special purpose. To find the best result, the experiment was repeated with different CNN parameters. The adjusted values of major parameters are as follows. Each CNN consisted of 2 convolution layers and 2 max-pooling layers. The window sizes of the convolution layer and the pooling layer were 3 and 2, respectively. The training epoch was 100.

**Table 1.** CNN feature extractors’ data set.

CNN Name	Class	Data
CNN_A	two-class	all original, all forged
CNN_B	N-class	all original
CNN_C	two-class	subject’s original, others’ original
CNN_D	two-class	subject’s original, forged for the subject

Each subject model was built with the subject’s 10 original signatures. The AE consisted of an input layer of 128 nodes, hidden layers, and an output layer of 128 nodes. The node activation function was ReLU [26] and training optimizer was RMSprop [27]. In the test phase, we conducted three types of signature verification tests and the test data configuration is shown in Table 2. For the base result, the adjusted values of major AE parameters are as follows. The number of hidden layers was 3 and the number of nodes were 30, 20, and 30 for each layer. The training epoch was 500.

**Table 2.** Signature verification test data for each subject.

Verification Test	Class 0 (True)	Class 1 (False)
Subject/Others	10 (original)	20 × 19 others (original)
Time difference	20 (time-deferred)	20 × 9 others (original)
Skilled forgery	10 (original)	20 (forged for the subject)

The equal error rate (EER) is used as an evaluation metric for signature verification. All machine-learning models were implemented using the Theano [28] library, which is a well-known open source machine-learning library.

#### 4.2. Signature Verification Results

Table 3 shows the signature verification results of the proposed method and a comparison with the results of a previous work [3].

**Table 3.** CNN-AE classification comparison with EER.

Models	Subject/Others	Time Difference	Skilled Forgery
MLP [3]	3.8%	7.7%	18.1%
AE	3.5%	9.1%	13.7%
CNN_A-AE	18.1%	23.7%	4.3%
CNN_B-AE	9.1%	11.3%	23.3%
CNN_C-AE	4.1%	4.5%	4.6%
CNN_D-AE	3.5%	4.8%	3.2%
<b>CNN_AC-AE (The Proposed Scheme)</b>	<b>2.7%</b>	<b>4.5%</b>	<b>4.4%</b>

As mentioned above, AE (one-class model) showed a better result for distinguishing skilled forgery than MLP (two-class model), but not for others. In distinguishing forgery, CNN\_D-AE that was trained with the forged signatures of the subject showed the best accuracy (3.2%). Although this method could not be practically applied, this result showed that a specially trained CNN could learn characteristics for that purpose, as we expected. The CNN\_A-AE that was trained with all original and all forged signatures showed the second-best accuracy (4.3%). However, it did not perform

well in the subject/others and time difference tests. CNN\_C-AE showed better performances in the subject/others (4.1%) and time difference (4.5%) tests, but it did not perform well in the skilled forgery test compared to CNN\_A-AE. Therefore, we tried a combination of CNN\_A and CNN\_C (called CNN\_AC-AE) by concatenating *S-vector* from both networks and doubling the size of the AE input/output. The performance of the CNN\_AC-AE was better than that of CNN\_C-AE in the skilled forgery and subject/others tests, and was the same in the time difference test. It showed 1.1% (subject/others), 3.2% (time difference), and 13.7% (skilled forgery) better results than those of the previous work [3] and showed 1.8%, 4.6%, and 9.3%, respectively, better performances than those of the test using AE only.

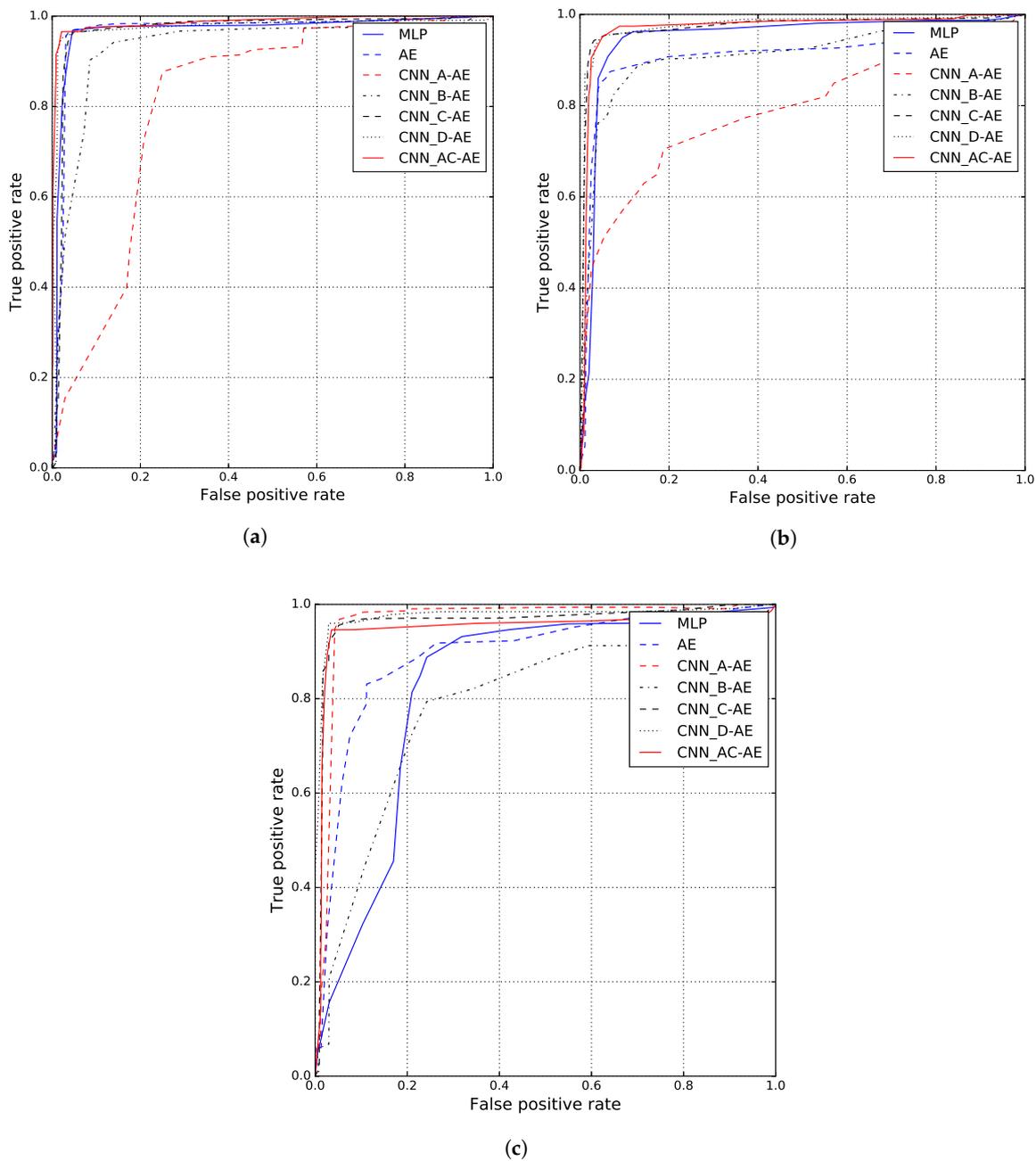


Figure 8. ROC curve of: (a) subject/others; (b) time difference; and (c) skilled forgery.

Figure 8 shows the experiment results as receiver operation characteristic (ROC) curves. An ROC curve refers to a graphical plot that illustrates the correlation between sensitivity and specificity of a classifier system. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR). Therefore, the closer a curve is to 1, the more accurate is the corresponding classifier. Figure 8a shows ROC curves for subject/others signature authentication. The CNN\_A-AE showed the worst accuracy and the CNN\_AC-AE showed the best performance. Because the CNN\_A-AE was trained with the forged signatures, which are similar to those of the subject, it has a higher probability of misclassifying subject signatures as forged signatures. Figure 8b shows the ROC curves for the time difference test. The worst result was obtained by the CNN\_A-AE and the best one was obtained by the CNN\_AC-AE. The CNN\_A-AE has higher probability of misclassifying subject signatures changed over time as forged signatures for the same reason mentioned above. In the results of distinguishing forgery in Figure 8c, MLP, AE, and CNN\_B-AE showed worse accuracy than others. MLP exposed a weakness in the skilled forgery test due to being a 2-class model. AE and CNN\_B-AE did not perform well. This result shows that the *S-vector* generated by the trained CNN is more effective than raw signature data. The CNN\_D-AE showed the best result and the CNN\_AC-AE showed the second-best accuracy. However, note that the CNN\_D-AE is not a practical model.

## 5. Discussions

CNN is widely known as an outstanding image classification algorithm. In this study, we used CNN as a feature extractor of signatures, i.e., time series data. Our main assumption was that CNN could extract effective features from signatures as well as from images. The comparison results between the AE model and the proposed scheme (CNN\_AC-AE model) show that, at least, the *S-vector* from a CNN feature extractor is valuable for signature verification.

In realistic environments, users could sign on a smartphone lying on a table as well as while holding it in their hands. In the case of a smartphone lying on the table, the changes in accelerometer values are slight, which negatively affects the verification accuracy. We therefore surveyed 50 smartphone users and found that 47 of them use a smartphone by holding it in their hand all the time except while watching videos and playing games. Moreover, when constructing the test set, although we did not mention the position of the smartphone, all subjects signed on the smartphone by holding it in their hands.

In the training phase, the training epoch depends on the size, amount, and type of data. The size of our test set is relatively smaller than that of general machine learning datasets. The epochs of CNN and AE are determined as 100 and 500, respectively, via many trials.

As a signature is generated by human action, mistakes occur frequently, even though the signature is drawn by the same person. In the test phase, a mistake may result in a single verification failure. However, in the training phase, a mistake could lead to a decrease in verification accuracy. We therefore provided a cancel function in the application for the signature collection process, that allows a user to cancel a signature just drawn if he or she made a mistake. The cancel function is very simple but effective for signature verification. In relation to this practical issue, our previous work [29] proposed automatic data filtering for signing mistakes. Prior to the training phase, it compares sample signatures, eliminates out-of-range signatures, and uses only the remaining as training data. For fair comparison, we do not apply the automatic data filtering in this study.

## 6. Conclusions and Future Work

This paper proposed a new feature-extraction method using CNN for dynamic signature verification. A CNN was exploited to generate an *S-vector* as the input to the autoencoder. Experimental results showed that the CNN trained with forged and genuine signatures could generate an *S-vector* that achieved good accuracy in distinguishing forged signatures but did not perform well in other tests, namely, subject/others and time difference. Therefore, we combined the *S-vector* of two CNN feature extractors that were trained for different purposes, which yielded good results in all the three types of

tests. Using the proposed method, the EER was improved by up to 1.1% (subject/others), 3.2% (time difference), and 13.7% (skilled forgery) compared to that in the previous work. The proposed dynamic signature verification system could be used for secondary authentication such as in smartphones, PDAs, mobile computers, financial settlements, and mobile banking.

In future work, we plan to verify signatures drawn by a hand gripping a smartphone, i.e., a user draws a signature in the air by a hand gripping a smartphone, and not on a screen by a finger. We expect that the large signature drawn by hand could have positive effects of verification accuracy and user convenience. However, the features and characteristics of the hand-drawn signature might be different from those of a finger-drawn signature. The differences could bring up some issues.

**Acknowledgments:** This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (No. 2016-0-00110, Development of Information Security Core Technology and 2016-0-00173, Security Technologies for Financial Fraud Prevention on Fintech).

**Author Contributions:** The authors contributed equally to this work.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Roh, J.H.; Lee, S.H.; Kim, S. Keystroke Dynamics for Authentication in Smartphone. In Proceedings of the International Conference Information and Communication Technology (ICTC), Jeju Island, Korea, 19–21 October 2016; pp. 1155–1159.
2. Liu, D.; Dong, B.; Gao, X.; Wang, H. Exploiting Eye Tracking for Smartphone Authentication. In Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS), New York, NY, USA, 2–5 June 2015; pp. 457–477.
3. Buriro, A.; Crispo, B.; Delfrari, F.; Wrona, K. Hold and Sign: A Novel Behavioral Biometrics for Smartphone User Authentication. In Proceedings of the IEEE Symposium on Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 276–285.
4. Feng, H.; Wah, C.C. Online signature verification using a new extreme points warping technique. *Pattern Recognit. Lett.* **2003**, *24*, 2943–2951.
5. Fierrez-Aguilar, J.; Nanni, L.; Lopez-Peñalba, J.; Ortega-Garcia, J.; Maltoni, D. An on-line signature verification system based on fusion of local and global information. In Proceedings of Audio- and Video-Based Biometric Person Authentication (AVBPA), Rye Brook, NY, USA, 20–22 July 2005; pp. 523–532.
6. Nanni, L. An advanced multi-matcher method for on-line signature verification featuring global features and tokenised random numbers. *Neurocomputing* **2006**, *69*, 2402–2406.
7. Guru, D.S.; Prakash, H.N. Online signature verification and recognition: An approach based on symbolic representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1059–1073.
8. Gruber, C.; Gruber, T.; Krinninger, S.; Sick, B. Online signature verification with support vector machines based on LCSS kernel functions. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2010**, *40*, 1088–1100.
9. Iranmanesh, V.; Ahmad, S.M.S.; Adnan, W.A.W.; Malallah, F.L.; Yussof, S. Online signature verification using neural network and Pearson correlation features. In Proceedings of the IEEE Conference on Open Systems (ICOS), Subang, Selangor, Malaysia, 26–28 October 2014; pp. 18–21.
10. Fayyaz, M.; Saffar, M.H.; Sabokrou, M.; Hoseini, M.; Fathy, M. Online signature verification based on feature representation. In Proceedings of the International Symposium on Artificial Intelligence and Signal Processing (AISP), Mashhad, Iran, 3–5 March 2015; pp. 211–216.
11. Sae-Bae, N.; Memon, N. Online signature verification on mobile devices. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 933–949.
12. Antal, M.; Szabó, L.Z. On-line verification of finger drawn signatures. In Proceedings of the International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 12–14 May 2016; pp. 419–424.
13. Paudel, N.; Querini, M.; Italiano, G.F. Handwritten Signature Verification for Mobile Phones. In Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP), Rome, Italy, 19–21 February 2016; pp. 46–52.

14. Zhang, C.; Gao, W.; Song, J.; Jiang, J. An imbalanced data classification algorithm of improved autoencoder neural network. In Proceedings of the IEEE International Conference on Advanced Computational Intelligence (ICACI), Chiang Mai, Thailand, 14–16 February 2016; pp. 95–99.
15. Nguyen, T.V.; Sae-Bae, N.; Memon, N. Finger-drawn PIN Authentication on Touch Devices. In Proceedings of the IEEE International Conference on Image Process (ICIP), Paris, France, 27–30 October 2014; pp. 5002–5006.
16. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1998; pp. 255–258, ISBN 0-262-51102-9.
17. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
18. Marchi, E.; Vesperini, F.; Eyben, F.; Squartini, S.; Schuller, B. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 19–24 April 2015; pp. 1996–2000.
19. Nam S.S. Mobile Finger Signature Verification Robust to Skilled Forgery. *J. Korea Inst. Inf. Secur. Cryptol.* **2016**, *26*, 1161–1170. (In Korean)
20. Liu, Y.; Yang, Z.; Yang, L. Online signature verification based on DCT and sparse representation. *IEEE Trans. Cybern.* **2015**, *45*, 2498–2511.
21. Fischer, A.; Diaz, M.; Plamondon, R.; Ferrer, M.A. Robust score normalization for DTW-based on-line signature verification. In Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), Nancy, France, 23–26 August 2015; pp. 241–245.
22. Wijesoma, W.S.; Mingming, M.; Yue, K.W. On-line signature verification using a computational intelligence approach. In Proceedings of the Fuzzy Days, Dortmund, Germany, 1–3 October 2001; pp. 699–711.
23. Ryu, S.Y.; Lee, D.J.; Chun, M.G. A Robust On-line Signature Verification System. *Int. J. Fuzzy Log. Intell. Syst.* **2003**, *3*, 27–31.
24. Fierrez, J.; Ortega-Garcia, J.; Ramos, D.; Gonzalez-Rodriguez, J. HMM-based on-line signature verification: Feature extraction and signature modeling. *Pattern Recognit. Lett.* **2007**, *28*, 2325–2334.
25. Blunsom, P.; Grefenstette, E.; Kalchbrenner, N. A convolutional neural network for modelling sentences. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Baltimore, MD, USA, 22–27 June 2014.
26. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), Stateline, NV, USA, 3–6 December 2012; pp. 1097–1105.
27. Dauphin, Y.; de Vries, H.; Bengio, Y. Equilibrated adaptive learning rates for non-convex optimization. In Proceedings of the Conference on neural information processing systems (NIPS), Montréal, QC, Canada, 7–12 December 2015; pp. 1504–1512.
28. Theano 1.0 Release. Available online: <http://deeplearning.net/software/theano/> (accessed on 2 December 2017).
29. Nam, S.; Park, H.; Seo, C.; Choi, D. Customized Data Filtering For Mobile Signature Verification. *Int. J. Appl. Eng. Res.* **2017**, *12*, 8727–8731.

