Convolutional Auto-Encoder Model for Finger-Vein Verification

Borui Hou
School of Instrument Science and Engineering
Southeast University
Nanjing, China
Email: boruihou@seu.com.cn

Ruqiang Yan*
School of Instrument Science and Engineering
Southeast University
Nanjing, China
Corresponding author: ruqiang@seu.edu.cn

Abstract—This paper presents a novel deep learning-based method that integrates a Convolutional Auto-Encoder (CAE) with support vector machine (SVM) for finger vein verification. The CAE is used to learn the features from finger vein images and the SVM is used to classify finger vein from these learned feature codes. The CAE consists of a finger vein encoder, which extracts high-level feature representation from raw pixels of the images, and a decoder which outputs reconstruct finger vein images from high-level feature code. As an effective classifier, support vector machine (SVM) is introduced in this paper to classify the feature code which is obtained from CAE. Experiments prove that the proposed deep learning-based approach has superior performance in learning features than traditional method without any prior knowledge, presenting a good potential in the verification of finger vein.

Keywords—Biometrics; finger-vein; deep learning; convolutional auto-encoder ; support vector machine.

I. INTRODUCTION

Automatic personal verification using biometrics has received widespread attention and increasing demand for border control, access control, attendance systems and financial security [1]. Biometrics is a highly safe and reliable technique, which utilizes physical or physiological characteristics to perform personal verification. Up to data, various types of biometrics have been proposed, including face [2,3], iris [4,5], fingerprint [6], palm print [7], posture [8], ECG [9,10], finger vein [11] and so on.

As an intrinsic biometric method, finger vein is much harder to forge as it is located inside the subcutaneous layer [12]. As compared to other biometric technologies, finger vein verification has some advantages: (1) Fast: small image data for fast processing (2) Non-contact: not affected by skin conditions. (3) Biometric: finger vein patterns can be identified on the living body. (4) High safety: vein pattern is internal features, not easy to replicate [13].

Like other biometric verification method, a finger-vein-based verification method mainly has four steps, data acquisition, data pre-processing, feature extraction and data verification. Usually, data pre-processing consists of several steps and they are image normalization, region of interest (ROI) extraction and Image enhancement. Finger vein verification methods can be divided into two categories: template matching and feature classification. In data pre-processing, various operations on finger vein are performed such that finger vein structure is enhanced and ROI images are extracted from the original images [14]. After feature extraction, suitable features from the veins are extracted and then verified in data verification.

In data pre-processing, various filters and transforms has been introduced in finger vein enhancement, such as Gabor filter [15,16], and Curvelet transform [17]. Gabor filters was introduced in finger-vein image enhancement and then the Phase-Only-Correlation (POC) measure method was used to verify finger-vein image [15]. While in [16], a Gabor filter-based finger-vein feature extraction approach was proposed, where a set of Gabor filters was introduced to exploit the finger-vein features at different orientations and scales, and then a finger-vein code was constructed through these features for identification. Based on Curvelet transform, a multiscale enhancement method was used to perform on the finger-vein image, and then a local interconnection structure neural network is applied to identify finger-vein images [17]. However, these enhancement methods are not self-adaptive to variable width veins, while noise also gets enhanced when it presents closely resembles the vein structure. As a self-adaptive enhancement method, a repeated line tracking method based on valley feature detection was proposed to extract the vein network by calculating difference between the center pixel value and the pixel value in the corresponding range [18]. In [19], the repeated line tracking method has been modified by calculating the maximum curvature information during vein tracking. Furthermore, a sliding window-based image enhancement method was proposed to enhance finger vein network [20], and such a method was improved using a dual-sliding window [21]. But it is still observed that thin veins are not effectively enhanced due to small width.

After image enhancement, features were extracted for data verification, such as random transform [12], local line binary pattern [22], Pattern map [23], PCA [24,25] and hybrid methods [26, 27]. However, random transform was used to extract features and then used as input to an artificial neural network classifier for image identification. A finger-vein pattern matching approach based on Local Line Binary Pattern (LLBP) was proposed in [22], where the LLBP features were extracted and then the matching scores were calculated by Hamming distance. Pattern map based on pixel-pattern-based texture feature (PPBTF), together with principal component analysis (PCA), was also applied for finger-vein recognition [23], where the PPBTF was used to represent finger vein pattern information, and PCA was...
introduced to further reduce the dimension of the PPBTFs before sending to a nearest neighbor classifier. As a popular dimensionality reduction and feature extraction technology, PCA [24] and (2D)² PCA [25] also introduced in finger vein feature extraction and then trained a neural network for verification, which result in a high recognition rate. Finger vein feature extraction was also accomplished by various ways. Combined PCA and LDA, a hybrid finger vein recognition method was proposed in [26], PCA and LDA were applied to extract features and SVM model was trained for recognition. In [27], local binary pattern was introduced to extract local features of finger vein images while wavelet transform was used to collect global features. However, these feature extraction approaches need to assume shape and structure of the finger-vein conform to a particular pattern, such as the distribution of valleys and segments, and it is not easy to design an effective finger vein feature description model.

In the field of finger vein verification, deep learning-based approach has been successfully applied in recent years. It is formed by deep neural network that provides powerful image processing capabilities without any prior knowledge, and has good performance in noisy images handling and feature representation learning adaptively. For example, a deep convolutional neural network (CNN) was proposed to design a new finger vein verification method, named as Deep-Vein, which showed a good effect on finger vein pattern matching [28]. The CNN was also used to build a deep-learning based segmentation model, which can extract robust vein patterns and reduce error rate of the finger vein verification [29]. To improve performance of finger vein verification, various CNN architectures and loss functions were investigated and achieved better performance over other finger-vein verification method [30]. A summary results of finger vein verification methods mentioned above are shown in Table 1.

Inspired by these prior efforts and our previous research [31], this paper presents a convolutional auto-encoder (CAE)-based deep learning model for finger-vein verification. The CAE network is new designed and applied to learning feature codes following a certain distribution, which are then processed by SVM for classification. The size of the extracted deep feature codes has been greatly decreased in the proposed method. And the method proposed in this paper has great improvement in two different databases. The remainder of the paper is organized as follows: Section II introduces the theoretical background of CAE and SVM. Then the proposed method is discussed in detail in section III, followed by experimental verification in section IV. Some conclusions are finally drawn in section V.

### TABLE 1 SUMMARY OF FINGER VEIN VERIFICATION METHODS IN THE LITERATURE WITH THIS WORK

<table>
<thead>
<tr>
<th>Ref</th>
<th>Method</th>
<th>Database</th>
<th>Two session</th>
<th>Database fingers</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Samples per each</td>
</tr>
<tr>
<td>[1]</td>
<td>Gabor filters with morphological method</td>
<td>Public</td>
<td>Yes</td>
<td>105²12</td>
<td>4.91%</td>
</tr>
<tr>
<td>[14]</td>
<td>Multi-scale matched filter + Line tracking</td>
<td>Public</td>
<td>Yes</td>
<td>105²12</td>
<td>4.47%</td>
</tr>
<tr>
<td>[18]</td>
<td>Repeated line tracking</td>
<td>Proprietary</td>
<td>No</td>
<td>678*</td>
<td>0.145%</td>
</tr>
<tr>
<td>[19]</td>
<td>Maximum curvature</td>
<td>Proprietary</td>
<td>No</td>
<td>678*</td>
<td>0.0009%</td>
</tr>
<tr>
<td>[27]</td>
<td>LBP + Wavelet</td>
<td>Proprietary</td>
<td>No</td>
<td>50²8</td>
<td>0.01%</td>
</tr>
<tr>
<td>[28]</td>
<td>CNN</td>
<td>Public</td>
<td>No</td>
<td>1000²5</td>
<td>0.42%</td>
</tr>
<tr>
<td>[30]</td>
<td>CNN with Triplet similarity loss</td>
<td>Public</td>
<td>Yes</td>
<td>105²12</td>
<td>1.41%</td>
</tr>
<tr>
<td></td>
<td>Supervised discrete hashing with CNN</td>
<td>Public</td>
<td>Yes</td>
<td>105²12</td>
<td>2.14%</td>
</tr>
<tr>
<td>[31]</td>
<td>CAE + CNN</td>
<td>Public</td>
<td>Yes</td>
<td>492²12</td>
<td>0.16%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Public</td>
<td>No</td>
<td>636²6</td>
<td>6.28%</td>
</tr>
<tr>
<td>This paper</td>
<td>CAE + SVM</td>
<td>Public</td>
<td>Yes</td>
<td>492²12</td>
<td>0.12%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Public</td>
<td>No</td>
<td>636²6</td>
<td>0.21%</td>
</tr>
</tbody>
</table>

### II. THEORETICAL BACKGROUND

#### A. Convolutional Neural Networks

Successfully applied in the field of computer vision, convolutional neural network has demonstrated its powerful ability to represent features. It uses multiple filters which shares different parameters to extract image features. In general, CNN includes two kinds of connections: convolution and pooling as illustrated in fig. 1.
For convolution connection, the features are extracted by performing a two-dimensional convolution of the input map and the convolutional kernel. Here, the map refers to an image with multiple channels. Let \( x^m_i \) represent the \( m \)-th channel of input map at layer \( i \), the output map \( y^n_i \) at \( n \)-th layer at \( i \) can be represented as

\[
y^n_i = ReLU\left( \sum_{m}^{M-1} \omega^n_{i,m} \ast x^m_i + b^n_i \right)
\]  

(1)

where \( \omega^n_{i,m} \) is the convolutional kernel between the \( y^n_i \) and \( x^m_i \), the symbol \( \ast \) represents convolutional operation, \( M-1 \) is the number of input maps, and the bias of the \( n \)-th output map is \( b^n_i \).

As for the activation function, Rectified Linear Units (ReLU) (\( y = \max(0, x) \)) is chosen in Eq. (1). The rectified linear units (ReLU) instead of traditional sigmoid units is introduced as the activation function of hidden layers, because the ReLU works better on capturing patterns in natural images and improves the ability of neural network on solving image de-noising problem.

In pooling connection, the filter response can be reduced to lower dimensions and the input representation can be more compact. In general, there are two kinds of operation: max-pooling and averaging-pooling. Max-pooling decreases the dimension of data simply by taking only the maximum data, while average-pooling works in a similar way by taking the average of inputs instead of maximum. Based on the conceptual difference between the two methods, max-pooling is sensitive to texture information of the image, and average-pooling retains more background information of the image. Therefore, max-pooling is more conducive to extract the feature information of the image. In this study, max-pooling is employed after the ReLU output is calculated.

B. Convolutional Auto-Encoder

An auto-encoder consists of two parts: encoder and decoder as illustrated in Fig. 2. The encoder converts the input \( x \) to a hidden representation \( y \) (feature code) using a deterministic mapping function. Typically, it is an affine mapping function followed by non-linearity:

\[
y = f(Wx + b)
\]  

(2)

where, \( W \) is weight between input \( x \) and hidden representation \( y \) and \( b \) is bias.

The decoder implements the process of reconstructing the output \( z \) by \( y \), which can be expressed as:

\[
z = f'(W'y + b')
\]  

(3)

where \( W' \) is weight between hidden representation \( y \) and output \( z \) and \( b' \) is bias. Similar to the input \( x \), \( z \) is considered as the reconstruction of \( x \).

The principle of training an auto-encoder is to minimize the reconstruction error, which can be realized by minimizing the following cost function \( J_{AE} \):

\[
J_{AE} = \frac{1}{p} \sum_{i=1}^{p} L[x_i, z_i]
\]  

(4)

where \( p \) is the number of input images, \( x_i \) is the \( i \)-th input image, and \( z_i \) is the reconstructed image corresponding to \( x_i \). \( L[x_i, z_i] \) represents reconstruction error of the input image \( x_i \), which can be measured by mean square error or cross entropy. In this study, the mean square error between the input image \( x_i \) (\( i = 1, 2, \ldots p \)) and the reconstructed patch of image \( z_i \) is used. Correspondingly, \( L[x_i, z_i] \) can be expressed as:

\[
L_{AE}[x_i, z_i] = \| x_i - z_i \|^2
\]  

(5)

Convolutional auto-encoder combines the local convolution connection with the auto-encoder, which is a simple step that adds convolution operation to inputs. Correspondingly, a convolutional auto-encoder is consisted of convolutional encoder and convolutional decoder. The convolutional encoder realizes the process of convolutional conversion from the input to feature maps, while convolutional decoder implements the convolutional conversion from feature maps to the output. In CAE, the extracted features and the reconstructed output are calculated through CNN. Thus, Eq. (2) and Eq. (3) can be rewritten as:

\[
y = ReLU(\omega x + b)
\]  

(6)

\[
z = ReLU(\omega'y + b')
\]  

(7)

where \( \omega \) represents the convolutional kernel between the input and the code \( y \), \( \omega' \) represents the convolutional kernel between the code \( y \) and the output. \( b \) and \( b' \) are bias. Moreover, the parameters of the encoding and decoding operations can be computed using unsupervised greedy training.

C. Support Vector Machine

SVM, which is derived from learning theory, has been widely applied and has the advantage in automatic complexity control to avoid over-fitting [16]. The main conception of SVM is to find a hyperplane in a high-dimensional space by maximizing the minimum distance between the hyperplane and training...
samples or by separating the training samples of each class. Originally designed for binary classification, one-against-one SVM training scheme is proposed to deal with the multi-class classification problem. The multiclass classification problem can be decoupled to several two-class classification problems and a voting strategy is introduced. Every binary classification is considered to be a voting and the maximum number of votes decides the class of sampling. The SVM is defined as:

$$f(y_{\text{code}}) = \text{sign}[\alpha^T \varphi(y_{\text{code}}) + b]$$  \hspace{1cm} (8)

where $\alpha$ is weight vector and $b$ is a bias, and $\varphi(y_{\text{code}})$ represents the feature vector that maps feature $y_{\text{code}}$ into the high dimension feature space. Given a training sample $(y_{\text{code}}^i, L_{\text{code}}^i)$, the function margin can be defined as:

$$\hat{\gamma}^i = L_{\text{code}}^i(a^T y_{\text{code}}^i + b)$$  \hspace{1cm} (9)

In order to find the maximum geometric margin $\gamma$, the following optimization problem is proposed:

$$\max_{\alpha, b, \|\alpha\|} \hat{\gamma} \frac{1}{\|\alpha\|}$$  

s.t. $L_{\text{code}}^i(a^T y_{\text{code}}^i + b) \geq \hat{\gamma}, i = 1, ..., m$  \hspace{1cm} (10)

subject to scaling constraint, $\hat{\gamma} = 1$. Plugging this into our problem above, and maximizing $\hat{\gamma} / \|\alpha\| = 1 / \|\alpha\|$ , the optimization problem is shown as:

$$\min_{\alpha, b, \|\alpha\|} \frac{1}{2} \|\alpha\|^2$$  

s.t. $L_{\text{code}}^i(a^T y_{\text{code}}^i + b) \geq 1, i = 1, ..., m$  \hspace{1cm} (11)

Then the Lagrangian function can be constructed to solve the optimization problem in Eq. (12):

$$L(\alpha, b, \alpha_t) = \frac{1}{2} \|\alpha\|^2 - \sum_{i=1}^{m} \alpha_t [L_{\text{code}}^i(a^T y_{\text{code}}^i + b) - 1]$$  \hspace{1cm} (12)

where $\alpha_t$ is Lagrangian multipliers. After solving the Eq. (12), the SVM classifier can be written as:

$$f(x) = \text{sign}\left[ \sum_{i=1}^{m} \alpha_t K(y_{\text{code}}, y_{\text{code}}^i) + b \right]$$  \hspace{1cm} (13)

where $K(y_{\text{code}}, y_{\text{code}}^i)$ is a kernel function. Through using the kernel function, SVMs can learn in the high dimensional feature space without having to explicitly find feature vector $\varphi(y_{\text{code}})$. There are three basic SVM kernels, including linear, poly and RBF. Linear kernel is the simplest kernel function, Poly kernel is a non-stationary kernel, and the polynomial degree is 2 in this paper. The RBF kernel, radial basis function kernel, is the most widely used kernels and usually has good performance. The kernel functions are shown as:

Linear kernel: $K(x, y) = x^T y + c$

Poly kernel: $K(x, y) = (\alpha x^T y + c)^2$

RBF kernel: $K(x, y) = \exp(-\gamma \|x - y\|^2)$  \hspace{1cm} (14)

where, $x, y$ refer to $y_{\text{code}}$ in this paper, the $c$ is an optional constant, $\alpha$ and $y$ are adjustable parameters. In the experiments, three basic SVM kernels have been used, and the performance is evaluated based on classification accuracy and equal error rate.

### III. METHODOLOGY

By taking advantage of CAE and SVM, a CAE-based deep feature learning integrated with SVM for finger vein verification is proposed in this paper, where the CAE network is used to learn features from the finger vein image, and then sent to SVM classifier to identify different finger veins. The flowchart of the proposed method is shown in Fig. 3.

---

**Fig. 3.** The flowchart of CAE–based deep feature learning integrated with SVM for finger vein verification.
Specifically, the finger vein image $x$ is inputted to the CAE. Convolution operation is used to extract the image features, and max pooling worked as down-sampling to obtain the feature code $y$, which is $m$-channel ($m$ is the number of convolutional kernels) probability maps, representing finger vein features with the values between 0 and 1. The output image $z$ is then reconstructed through a combination of convolution and upsampling operations. The target is to minimize the mean square error between the output image $z$ and the input image $x$. The smaller the loss function value is, the more similar the output image $z$ will be to the input image $x$. It also shows that the feature code $y$ can well represent the image $x$. After extracting finger vein features, SVM is used for classification. The input to the SVM comes from the feature code $y$. The SVM classifier has been widely used in pattern classification for same problem domain and achieved good results.

IV. EXPERIMENTAL STUDY

A. Datasets

Experimental study is conducted on two publicly available datasets to verify the performance of the proposed finger vein verification method. These experimental datasets are the FV_USM database and SDUMLA database.

Collected 123 subjects, 83 males and 40 females, the FV_USM dataset has a total of 5904 images in two sessions. There is an interval of longer than two weeks between two sessions. In each session, each finger has 6 images samples and thus 2952 images were obtained, while, the SDUMLA dataset collected 106 subjects, every subject has 6 images for 6 different fingers and totally 3816 images.

After image acquisition, ROI images were extracted from finger vein images. The ROI of the finger vein has been provided in FV_USM dataset, while in SDUMLA dataset the method mentioned in [14] has been introduced to extract the ROI image. For the purpose of reducing the computational cost, the image size is normalized to 48x144x3 pixels in this experiment for these two datasets. And pixel scale of the image is also normalized and contrast enhanced to eliminate the effects of uneven brightness. If a pixel belongs to the vein pattern, the gray level of the pixel is similar to other part of the same vein pattern. Through normalization and contrast enhancement, the finger vein details are more clearly which is important to training. Then data augmentation method is also applied in the experiments. Condition information, such as different orientation, location, scale, brightness, is added to the underlying data through data augmentation. In this paper, data augmentation methods, such as flip, rotation, shift, shear and zoom, were randomly introduced to these two datasets, and thus the training samples of each class has been increased after data augmentation. For FV-USM dataset, there are 123x4 classes and the number of samples of each class has been increased from 12 to 60, while there are 106x6 classes in SDUMLA datasets and the number of samples of each class has been increased from 6 to 60. Specific data augmentation parameters are shown in the Table 2. And images after data augmentation in two datasets are shown in Fig. 4.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Parameters</th>
<th>Flip True</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>Rotation (degree range)</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>Shift (Range scale)</td>
<td>Width 0.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Height 0.05</td>
<td></td>
</tr>
<tr>
<td>Shear range</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>Zoom range</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>

B. Parameter Selection for Network Structures

In order to obtain good verification performance, it is important to choose appropriate parameters for the proposed network structures. A wide range of experiments are conducted to test the effect of network structure parameters on the performance of finger vein verification. Two measures, RR and EER, are used to quantify the verification performance. RR means the rank-1 recognition rate of the dataset in this study. EER is the abbreviation of equal error rate, which is related to False Rejection Rate (FRR) and False Acceptance Rate (FAR). FRR is calculated by genuine scores while the FAR is computed by impostor scores. Eq. (15) and Eq. (16) show how FAR and FRR are computed.

$$\text{FAR} = \frac{\text{Number of matching scores in false acceptance}}{\text{Number of matching scores}}$$  \hspace{1cm} (15)

$$\text{FRR} = \frac{\text{Number of matching scores in false rejection}}{\text{Number of matching scores}}$$  \hspace{1cm} (16)
In practice, a threshold value ranging from 0 to 1 can be set for FAR and FRR calculation. In a given dataset, when the threshold value increases, the FAR will increase while FRR decreases. when FAR is equal to FRR, the error rate is EER.

In order to enhance the ability of representing features, deep convolutional neural network normally consists of multiple connected convolutional blocks. In this paper and [31], a typical convolutional block contains 2 convolutional layers, a pooling layer or up-sampling layer. For CAE structure in this paper, the filter number is set as 32, the convolution block layer number is 2 and convolution kernel size is set as 3. The size of the input image is normalized as 48 x 144. As a result, the feature code size for the SVM classifier is 24x72x32. These models are trained with the same initial parameters. The batch size is 32. An initial learning rate of 0.001 is applied to train the CAE. At initialization, kernels of each layers are randomly selected from a Gaussian distribution of zero mean and 0.01 standard deviations, while the bias are randomly set from a Gaussian distribution with a mean of 0.5 and a standard deviation of 0.01. Considering both performance and training time, Adam method is chosen as the optimizer, as it has excellent performances in improving the traditional gradient descent and promoting hyper parameter dynamic adjustment.

For CAE structure in [31], the filter number is also set as 32, the convolution block layer number is 2 and convolution kernel size is set as 3. As a result, the feature code size for the CNN classifier is 24x72x32. These models are trained with the same initial parameters. The batch size is also 32. The details of the CAE in this paper and in [31] are listed in Table 3. The loss curve of proposed CAE is shown in Fig 5.

TABLE 3 STRUCTURE PARAMETERS OF CAE

<table>
<thead>
<tr>
<th>Method</th>
<th>Layers</th>
<th>Kernel</th>
<th>Filters</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>Block</td>
<td>3</td>
<td>32</td>
<td>24x72x32</td>
</tr>
<tr>
<td></td>
<td>Features</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Decoder</td>
<td>Re_block</td>
<td>3</td>
<td>32</td>
<td>48x144x1</td>
</tr>
<tr>
<td></td>
<td>Block_1</td>
<td>3</td>
<td>32</td>
<td>24x72x32</td>
</tr>
<tr>
<td></td>
<td>Block_2</td>
<td>3</td>
<td>32</td>
<td>12x36x32</td>
</tr>
<tr>
<td></td>
<td>Block_3</td>
<td>3</td>
<td>64</td>
<td>6x18x64</td>
</tr>
<tr>
<td></td>
<td>Flatten</td>
<td>/</td>
<td>/</td>
<td>6912</td>
</tr>
<tr>
<td>This paper</td>
<td>Features</td>
<td>Dense</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Decoder</td>
<td>Re_block_1</td>
<td>3</td>
<td>64</td>
<td>12x36x32</td>
</tr>
<tr>
<td></td>
<td>Re_block_2</td>
<td>3</td>
<td>32</td>
<td>24x72x32</td>
</tr>
<tr>
<td></td>
<td>Re_block_3</td>
<td>3</td>
<td>32</td>
<td>48x144x3</td>
</tr>
<tr>
<td></td>
<td>2D Convolution</td>
<td>3</td>
<td>3</td>
<td>48x144x1</td>
</tr>
</tbody>
</table>

SVM kernels, including linear, poly (degree = 2) and RBF, are used to classify the feature code extracted by CAE model. Usually, linear kernels are often used to solve linear classification problems, while poly and RBF kernels usually adopted in learning of non-linear models. In order to discuss the effects of different penalty parameter c, experiments are conducted and the results are listed in Fig 6. In these experiments, the parameter c varies from 0.001 to 0.3, with a step size of 0.001, and the number of features is set as 32. Considering the performance, poly and RBF kernels are better than linear kernel, while the penalty parameter c of poly is 0.01 while the penalty parameter c of RBF is 0.012. The comparative receiver operating characteristic (ROC) using proposed method and three different kernels is shown in Fig 7. The results of classification performance on feature numbers are listed in Table 4. As shown in Table 4, the proposed method reached its best performance when the SVM kernel is poly. And the best performance is RR of 99.95%, EER of 0.12% (FV_USM) and RR of 99.78%, EER of 0.21% (SDUMLA). Comprehensive considering the performance of recognition rate and equal error rate in different kernels through cross-validation approach (as shown in Table 4 and Fig 7), poly kernel is selected in this paper.

TABLE 4 RESULTS OF CLASSIFICATION PERFORMANCE IN DIFFERENT PARAMETERS

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Features</th>
<th>SVM kernels</th>
<th>Linear</th>
<th>Poly</th>
<th>RBF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RR</td>
<td>EER</td>
<td>RR</td>
<td>EER</td>
</tr>
<tr>
<td>FV_USM</td>
<td>16</td>
<td>98.97</td>
<td>17.00</td>
<td>99.75</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>99.90</td>
<td>4.37</td>
<td><strong>0.12</strong></td>
<td>98.26</td>
</tr>
<tr>
<td>SDUMLA</td>
<td>16</td>
<td>98.93</td>
<td>18.39</td>
<td>99.75</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>99.69</td>
<td>12.13</td>
<td><strong>99.78</strong></td>
<td><strong>0.21</strong></td>
</tr>
</tbody>
</table>

Fig. 5 Loss curve of convolutional auto-encoder. (a) FV_USM; (b) SDUMLA

C. Test Result and Comparison with Other Methods

Using the structure parameters in the proposed method, recognition rate of 99.95%, equal error rate of 0.12% on the FV_USM dataset and recognition rate of 99.78%, equal error rate of 0.21% on the SDUMLA dataset can be achieved. As a comparison, results from some other methods including GCPA(Gabor+CLAHE+2DPCA) [15], sliding window-based method & Gabor filters (SWG) [20], Dual-sliding window & pseudo-elliptical transform (DWP) [21], Gabor wavelet coding (GWC) [32], Wide Line Detector &Pattern Normalization (WLD&PN) model [33], PCA [24], (2D)2PCA [25] and 2DPCA without offset [34] and are listed in Table 5. All of these methods are preprocessed including data pre-processing, feature extraction verification to get the best performance. It can be concluded that the method applied in this study can achieve the highest recognition rate with the highest rank-1 recognition rate and lowest equal-error rate among all the comparison methods. This verifies the validity of the proposed method for the verification of the finger vein.

![Fig. 6 Results of classification performance in different penalty parameters, (a) Equal error rate; (b) Recognition rate](image)

![Fig. 7 Comparative ROC using proposed method when number of features is 32, (a) FV_USM; (b) SDUMLA](image)

### Table 5: Rank-1 Recognition Rate (RR) and Equal Error Rate (EER) of Different Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>FV_USM</th>
<th>SDUMLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR(%)</td>
<td>EER(%)</td>
<td>RR(%)</td>
</tr>
<tr>
<td>GCPA [15]</td>
<td>90.62</td>
<td>5.72</td>
</tr>
<tr>
<td>SWG [20]</td>
<td>95.14</td>
<td>2.69</td>
</tr>
<tr>
<td>DWP [21]</td>
<td>97.02</td>
<td>2.32</td>
</tr>
<tr>
<td>GWC [32]</td>
<td>91.94</td>
<td>4.75</td>
</tr>
<tr>
<td>WLD&amp;PN [33]</td>
<td>96.51</td>
<td>2.37</td>
</tr>
<tr>
<td>PCA [24]</td>
<td>68.39</td>
<td>19.60</td>
</tr>
<tr>
<td>(2D)2PCA [25]</td>
<td>70.43</td>
<td>18.56</td>
</tr>
<tr>
<td>2DPCA [34]</td>
<td>67.78</td>
<td>20.57</td>
</tr>
<tr>
<td>CAE+CNN [31]</td>
<td>99.49</td>
<td>0.16</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>99.95</td>
<td>0.12</td>
</tr>
</tbody>
</table>

As for the time cost of the proposed method and other CNN based models, the proposed method has more time complexity than a direct CNN model as shown in Table 6. The directed CNN model which consists of the encoder mentioned in the proposed method and a softmax layer as output is used for comparison. Also, the time complexity of the proposed method in [31] is also listed in Table 6. For the proposed method, the feature extraction time is 2071ms and classification time is 153ms. In Table 6, computational platform used to run the algorithms is a PC desktop with the following characteristics (Intel Core Xeon(R) CPU E5-2609, 1.70 GHz x 16, RAM 256 GB, and GPU NVIDIA TITAN Xp 12GB).

### Table 6: The Time Complexity of the Proposed Method and Other CNN Based Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature Extraction Time(ms) + Classification Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAE+CNN [31]</td>
<td>3794 + 213</td>
</tr>
<tr>
<td>Encoder+softmax</td>
<td>1076</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>2070 + 153</td>
</tr>
</tbody>
</table>
V. CONCLUSION

In this study, a deep learning-based approach is presented in verification of finger vein. Data augmentation method is applied in this paper to increase the number and the diversity of data. The convolutional auto-encoder can effectively learn finger vein features, preserve the main information of the image, reduce redundant information, and improve the recognition efficiency with the help of SVM classifier. Experimental study on the FV_USM and SDUMLA dataset using the proposed method has shown that, in comparison with other methods in the literature, the proposed model has better performance, and can work more accurately and effectively. The EER of FVUSM improves from 0.16% [31] to 0.12%. The EER of SDUMLA improves from 6.28% [31] to 0.21%. The results of FVUSM doesn’t have much significant improvement because the results are quite good in [31]. But in SDUMLA database, there are significant improvement, and the result of EER has been greatly decreased. The information of the finger vein images has been further compressed, and it makes the proposed method more advantageous for practical applications. Further research can be extended to reduce the response time of finger vein verification when applied to a large database and data generative model will be introduced in data augmentation.

REFERENCES


