Chapter 5
Finger Vein Identification Using Convolutional Neural Network and Supervised Discrete Hashing

Cihui Xie and Ajay Kumar

Abstract Automated personal identification using vascular biometrics, such as from the finger vein images, is highly desirable as it helps to protect the personal privacy and anonymity in during the identification process. The Convolutional Neural Network (CNN) has shown remarkable capability for learning biometric features that can offer robust and accurate matching. We introduce a new approach for the finger vein authentication using the CNN and supervised discrete hashing. We also systematically investigate comparative performance using several popular CNN architectures in other domains, i.e., Light CNN, VGG-16, Siamese and the CNN with Bayesian inference-based matching. The experimental results are presented using a publicly available two-session finger vein images database. Most accurate performance is achieved by incorporating supervised discrete hashing from a CNN trained using the triplet-based loss function. The proposed approach not only achieves outperforming results over other considered CNN architecture available in the literature but also offers significantly reduced template size as compared with those over the other finger vein images matching methods available in the literature to date.

5.1 Introduction

Automated personal identification using unique physiological characteristics of humans, like face, fingerprint, or iris, is widely employed for e-security in a range of applications. In the past decade, there has been significant increase in the detection of surgically altered fingerprints, fake iris stamps, or the usage of sophisticated face masks, to thwart integrity of deployed biometrics systems. Vascular biometrics identification, like using finger vein patterns which are located at about three millimetres below the skin surface, can help to preserve the integrity of biometrics system as it is extremely difficult to surgically alter vascular biometrics. Another advantage of

C. Xie · A. Kumar (✉)
Department of Computing, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
e-mail: ajay.kumar@polyu.edu.hk

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_5

using finger vein biometrics is related to high degree of personal privacy as finger vein patterns are hidden underneath the skin surface and extremely difficult to acquire then covertly.

The possibility of personal identification using vascular patterns imaged using the light transmitted through hands was indicated in 1992 [6], but was not known to be demonstrated until 2000 [7]. Such earliest work demonstrated feasibility of finger vein identification using normalized-cross correlation. Miura et al. [12] later introduced repeated line tracking approach to improve the performance of finger vein identification, and they further enhanced the performance with maximum curvature [13]. Kumar and Zhou [8] introduced first publicly accessible finger vein database in public domain and comparatively evaluated a range of handcrafted features for the finger vein identification problem. The method introduced in [8] using Gabor filter-based enhancement and morphological operations is still regarded the best performing methods for matching finger vein images. A range of handcrafted features [2, 3, 8–13, 21], primarily obtained from the careful evaluation of the registered images, have been introduced in the literature to investigate finger-vein identification performance. Multiple features acquired from the two cameras [10] or using multiple feature extractors [22] can be combined to significantly improve performance for the vascular matching. One of the limitations of finger vein identification methods introduced in the literature is related to their large template size. Smaller template size is desirable to reduce storage and/or enhance the matching speed for the mobile and online applications. There have also been successful attempts to reduce the finger vein template size, like in [3, 9] or recently in [2] using sparse representation of enhanced finger vein images using the Gabor filters.

The finger vein matching methods available in the literature to date have judiciously introduced handcrafted features and demonstrated promising performance. However, the remarkable capabilities of the deep learning algorithms in automatically learning the most relevant vascular features are yet to be investigated or established. The objective of this work is to fairly investigate the effectiveness of self-learned features using popular convolutional neural network (CNN) architectures and develop more efficient and effective alternative for the automated finger vein identification. The experimental results and comparisons detailed in this chapter used light CNN [20], modified VGG-16 [16], CNN with Bayesian inference, and Siamese network with triplet loss function. Our reproducible [5] experimental results using *publicly* available database indicate that supervised discrete hashing in conjunction with CNN not only achieves outperforming results, but also significantly reduce the finger vein template size which offers increased matching speed. Table 5.1 in the following summarizes promising methods for the finger vein matching that have been introduced in the literature. This table also presents the template size in respective reference, which has been estimated from the details provided in respective reference, performance in terms of EER, and the database used for the performance evaluation. Reference [4] provides good summary of publicly available finger vein image databases introduced in the literature. The usage of two-session databases, particularly for the less constrained or contactless imaging setups as in [8], generates high intra-class variations and is highly desirable to generate fair evaluation of the matching algorithms

Table 5.1 Comparative summary of handcrafted finger vein features in the literature with this work

Ref.	Feature	Database	Two session	Template size (bytes)	EER	No. of subjects	No. of genuine scores	No. of impostor scores
[8]	Handcrafted (Even Gabor)	Public	Yes	106384	4.61%	105	1260	263,340
[8]	Handcrafted (Morphological)	Public	Yes	6710	4.99%	105	1260	263,340
[12]	Handcrafted (Repeated line tracking)	Proprietary	No	43200*	0.145%	678	678*	458,328*
[13]	Handcrafted (Maximum curvature)	Proprietary	No	43200*	0.0009%	678	678*	458,328*
[3]	Handcrafted (Local binary pattern)	Public	No	≤260*	3.53%	156	624	194,064
[2]	Handcrafted (Sparse representation using l1-norm)	Proprietary	No	630*	Unknown	17	Unknown	Unknown
[9]	Handcrafted (Extended local binary pattern)	Public	Yes	131328*	7.22%	105	1260	263,340
[21]	Handcrafted (Unknown algorithm)	Proprietary	Yes	20480*	0.77%	Unknown	10,000	499,500
[11]	Handcrafted (Histogram of salient edge orientation map)	Public	No	≤3496*	0.9%	100	3000	1,797,000
Ours	CNN with triplet similarity loss	Public	Yes	1024	13.16%	105	1260	263,340
Ours	Supervised discrete hashing with CNN	Public	Yes	250	9.77%	105	1260	263,340

*Computed by us from the details available in the respective reference

under more realistic usage/environments. Similarly, the usage of publicly available database can ensure reproducibility of results. Therefore, our all experiments in this chapter incorporate two-session and publicly available database from [8]. The last two rows in this table summarize best performing results from our investigation detailed in this work [19].

The rest of this chapter is organized as follows. Section 5.2 briefly describes on the preprocessing of the finger vein images and includes relevant steps for the image normalization, segmentation and enhancement. The CNN architectures, LCNN, VGG, LCNN with triplet similarity loss function, and LCNN with joint Bayesian formulation investigated in this are introduced in Sect. 5.3 while Sect. 5.4 details the supervised discrete hashing algorithm investigated to enhance performance and reduce the template size. The experimental results are presented in Sect. 5.4 and includes discussion on our findings, comparison with earlier methods. Finally, the key conclusions from this work are summarized in Sect. 5.5.

5.2 Image Normalization for Finger Vein Images

5.2.1 Preprocessing

Acquisition of finger vein images can introduce translational and rotational changes in different images acquired from the same finger or subject. Therefore, automated extraction of fixed region of interest (ROI) that can minimize such intra-class variations is highly desirable. The method of ROI localization considered in this work is same as detailed in [8] as it works well in most cases. Figure 5.1 illustrates samples of the acquired images using the near infrared camera.

Once the region of interest is localized, we can recover the binary masks corresponding to the ROI which can be used for alignment of finger vein samples, so that the adverse influence from the rotational changes in fingers can be minimized. The method for estimating the rotation is same as described in [8]. This estimated angle is used to align ROI, before the segmentation, in a preferred direction.

5.2.2 Image Enhancement

The finger vein details from the normalized images are subjected to the contrast enhancement to enhance clarity in vascular patterns which can be more reliable for

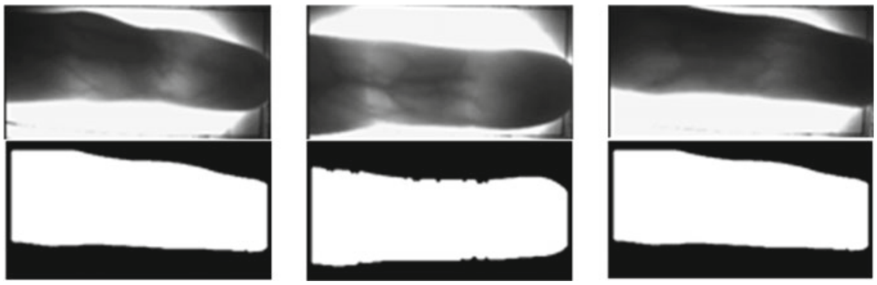

Fig. 5.1 Finger vein image samples before preprocessing (*first row*) and the binary images generated during the preprocessing (*second row*) stage

the training. Since vascular patterns are generally continuous, if a pixel belongs to the vascular pattern, there is a high possibility that its surrounding pixels are also part of the same vascular pattern and have similar gray level. Such observation is the same for nonvascular parts. Therefore, enhancement by computing the average gray level surrounding a pixel can help to enlarge the difference between the vascular parts and nonvascular parts, and makes the finger vein details more obvious as a result. After such enhancement, the vascular patterns become clearer with details as shown from sample images in Fig. 5.2.

The vascular patterns in the normalized image samples can be further enhanced by spatial filtering from orientation selective band pass filters, similar to as used in the enhancement of fingerprint images. We also attempted to ascertain usefulness of such enhanced finger vein images using the Gabor filters. These filters from the twelve different orientations are selected to generate enhanced finger vein images as shown in Fig. 5.4. Such enhanced images [8] using Gabor filters are effective in accentuating the vascular features and therefore its possible usage in automatically vascular features (Fig. 5.3) from CNN was also investigated in the experiments.

The finger vein image-processing operations essentially generates two kinds of enhanced images, ROI-A and ROI-B shown in Fig. 5.4, that were considered for the performance evaluation using the CNNs.

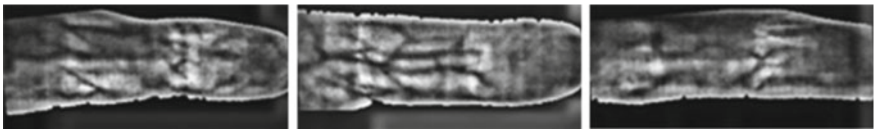

Fig. 5.2 Enhanced ROI vein images after rotation

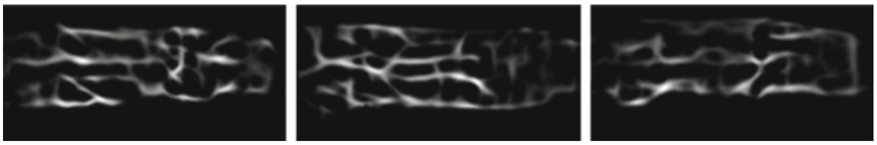

Fig. 5.3 Samples from even Gabor filtered finger vein images

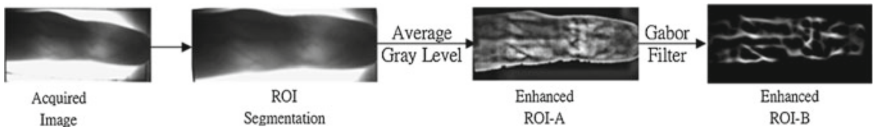

Fig. 5.4 Key steps in the generation of enhanced finger vein images for the CNNs

5.3 Convolutional Neural Network Architectures

A variety of models for the deep learning have been developed to learn useful feature representation but largely for the face biometric image patterns. A variety of such models using CNN have been introduced in the literature and were investigated to ascertain performance for the finger vein image matching. A brief introduction to various CNN architectures considered in this work is provided in the following sections.

5.3.1 Light CNN

The light CNN (LCNN) framework introduces a Max-Feature-Map (MFM) operation [20] between convolutional layers which establish a competitive relationship for superior generalization capability and reduce parameter space (compact feature representation). Such *maxout* activation (Fig. 5.5) function significantly reduces complexity and makes CNN lighter, where *conv* stands for convolutional layer.

For a convolutional layer without MFM, suppose that input size is $N_1 \times W_1 \times H_1$ and output size is $N_2 \times W_2 \times H_2$ then the required complexity using big 'O' notation can be represented as follows:

$$O\,(N_1 N_2 \Omega)\ \ where\ \Omega = W_1 \times H_1 \times W_2 \times H_2 \tag{5.1}$$

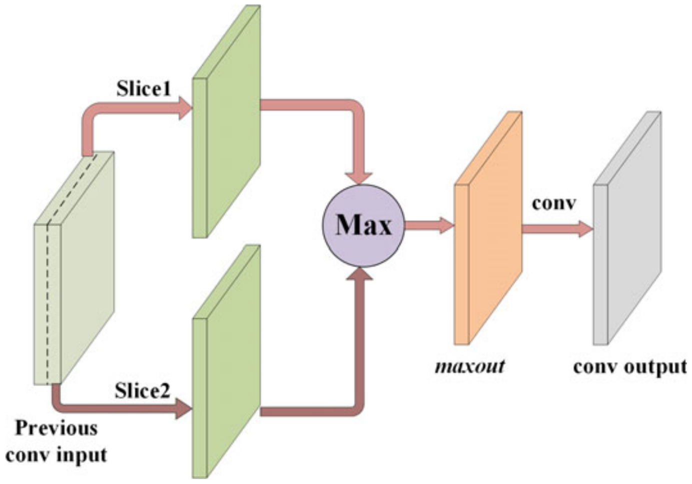

Fig. 5.5 Illustration for computing the Max-Feature Map in LCNN

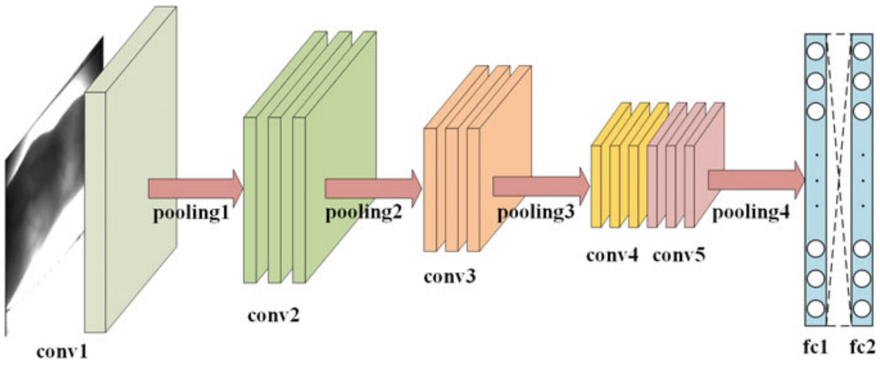

Fig. 5.6 The architecture for LCNN investigated in our experiments

For a convolutional layer with MFM, we could slice input into two equal-size parts, each with $\frac{N_1}{2} \times W_1 \times H_1$. Then for each corresponding element in two sliced parts we generate an output using *maxout* activation, which has a size of $\frac{N_1}{2} \times W_1 \times H_1$. With this smaller or lighter data as the input, complexity of convolutional layer reduces to

$$O\left(\frac{N_1 N_2 \Omega}{2}\right) \ where\ \Omega = W_1 \times H_1 \times W_2 \times H_2 \tag{5.2}$$

Comparing (5.1) and (5.2) we can infer that the usage of MFM can help to significantly reduce the complexity or make CNN lighter.

The loss function we used in this structure is *softmax* loss function. The basic idea is to combine *softmax* function with a negative log-likelihood, and the last layer information is used to estimate the identity of the class.

The architecture of LCNN employed in our experiments is shown in Fig. 5.6 (MFM part is excluded to maintain the clarity). This network contains 9 convolutional layers (conv), 4 pooling layers (pooling) and 2 fully connected layers (fc) and some assistant layers which are summarized in Table 5.2.

5.3.2 *LCNN with Triplet Similarity Loss Function*

Deep Siamese networks have been successfully incorporated to learn a similarity metric between a pair of images. We incorporated similar triplet similarity loss function as detailed in [14] for LCNN to learn the similarity metric.

We randomly select an image x^r from training set as *random* sample in Fig. 5.7. Then, we choose image x^p which is from the same class referred to as *positive* sample and image x^n which is from a different class referred to as *negative* sample. After LCNN, we get the features $f(x^r)$, $f(x^n)$, and $f(x^p)$. Our objective is to decrease the

Table 5.2 Details of layer information of LCNN

Index	Type	Filter size	Num	Stride	Pad
1	conv1	5	96	1	2
2	MFM1	–	48	–	–
3	pooling1	2	–	2	0
4	conv2a	1	96	1	0
5	MFM2a	–	48	–	–
6	conv2	3	192	1	1
7	MFM2	–	96	–	–
8	pooling2	2	–	2	0
9	conv3a	1	192	1	1
10	MFM3a	–	96	–	–
11	conv3	3	384	1	1
12	MFM3	–	192	–	–
13	pooling3	2	–	2	0
14	conv4a	1	384	1	1
15	MFM4a	–	192	–	–
16	conv4	3	256	1	1
17	MFM4	–	128	–	–
18	conv5a	1	256	1	1
19	MFM5a	–	128	–	–
20	conv5	3	256	1	1
21	MFM5	–	128	–	–
22	pooling4	2	–	2	0
23	fc1	–	512	–	–
24	MFMfc	–	256	–	–
25	fc2	–	500	–	–

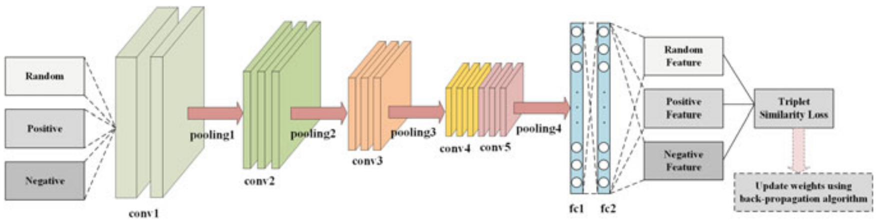

Fig. 5.7 The architecture for LCNN with triplet similarity loss function

similarity distance between *random* and *positive* features, and increase it between *random* and *negative* features, which indicates why it is named as triplet similarity loss. At the same time, we also need to ensure that there is a sufficient *margin* between them.

Suppose we have a random set $\mathbf{X^r} = \{x_i^r\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and its corresponding positive set $\mathbf{X^p} = \{x_i^p\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and negative set $\mathbf{X^n} = \{x_i^n\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$. Considering these notations, we can write our loss function as follows:

$$\sum\nolimits_{i=1}^{N} [\| f\left(x_i^r\right) - f\left(x_i^p\right) \|^2 - \| f\left(x_i^r\right) - f\left(x_i^n\right) \|^2 + margin]_+ \tag{5.3}$$

where $[\cdot]_+$ presents that we maintain positive values and change others to zero. The architecture of LCNN with such triplet similarity loss function is shown in Fig. 5.7 and detailed in Table 5.3. When the set of input consists of n random samples, n positives and n negatives, we generate $3n \times 500$ features. These pairs were split into three parts, each with the size of $n \times 500$, and used as the input for computing triplet similarity loss for updating the neuron weights during the network training.

5.3.3 Modified VGG-16

The Visual Geometry Group architecture with 16 layers (VGG-16) [16] was modified for the CNN to directly recover the match scores, instead of the feature vectors, in our experiment. Our modification was motivated to fit the rectangular finger vein ROI images without introducing the distortion. We used pair of images rather than single image as input in conventional VGG-16 since we want to compare the similarity between two finger vein images. The input image size is also different from conventional VGG-16, which is 224 × 224, while it's 128 × 488 pixels for our finger vein ROI images. The training phase utilized the cross-entropy loss function which can be written as follows:

$$-\frac{1}{n}\sum\nolimits_{i=1}^{n}\left[y_i \log\left(\widehat{\boldsymbol{y}}_{\boldsymbol{l}}\right) + (1 - y_i)\log\left(1 - \widehat{\boldsymbol{y}}_{\boldsymbol{l}}\right)\right] \tag{5.4}$$

where $\widehat{\boldsymbol{y}}_{\boldsymbol{l}} = g(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}})g(\cdot)$ is the logistic function, $\boldsymbol{x}_{\boldsymbol{i}}$ is the extracted feature and $\mathbf{w}$ is the weight that needs optimized during the training. The architecture of Modified VGG-16 (MVGG) is shown in Fig. 5.8 and Table 5.4.

5.3.4 LCNN with Joint Bayesian Formulation

The principal component analysis (PCA) is a classical method to extract the most important features and is popular for the dimensionality reduction of the features. In another set of experiments, we incorporated PCA for the dimensionality reduction of features extracted from LCNN and then employed joint Bayesian [1] approach as distance metrics for matching finger vein images.

For any feature $\boldsymbol{f}$ extracted from LCNN, we regard it as combination of two parts $\boldsymbol{\mu}$ and $\boldsymbol{\varepsilon}$ where $\boldsymbol{\mu}$ is the average feature of the class to which $\boldsymbol{f}$ belongs and $\boldsymbol{\varepsilon}$ is

Table 5.3 Details of layer information of LCNN with triplet similarity loss

Index	Type	Input index	Output index	Filter size	Output size	Stride	Pad
1	DataR	–	4	–	N*W*H	–	–
2	DataP	–	4	–	N*W*H	–	–
3	DataN	–	4	–	N*W*H	–	–
4	Data	1,2,3	5	–	3N*W*H	–	–
5	conv1	4	6	5	3N*96	1	2
6	MFM1	5	7	–	3N*48	–	–
7	pooling1	6	8	2	–	2	0
8	conv2a	7	9	1	3N*96	1	0
9	MFM2a	8	10	–	3N*48	–	–
10	conv2	9	11	3	3N*192	1	1
11	MFM2	10	12	–	3N*96	–	–
12	pooling2	11	13	2	–	2	0
13	conv3a	12	14	1	3N*192	1	1
14	MFM3a	13	15	–	3N*96	–	–
15	conv3	14	16	3	3N*384	1	1
16	MFM3	15	17	–	3N*192	–	–
17	pooling3	16	18	2	–	2	0
18	conv4a	17	19	1	3N*384	1	1
19	MFM4a	18	20	–	3N*192	–	–
20	conv4	19	21	3	3N*256	1	1
21	MFM4	20	22	–	3N*128	–	–
22	conv5a	21	23	1	3N*256	1	1
23	MFM5a	22	24	–	3N*128	–	–
24	conv5	23	25	3	3N*256	1	1
25	MFM5	24	26	–	3N*128	–	–
26	pooling4	25	27	2	–	2	0
27	fc1	26	28	–	3N*512	–	–
28	MFMfc	27	29	–	3N*256	–	–
29	fc2	28	30,31,32	–	3N*500	–	–
30	SliceR	29	33	–	N*500	–	–
31	SliceP	29	33	–	N*500	–	–
32	SliceN	29	33	–	N*500	–	–
33	Loss	30,31,32	–	–	–	–	–

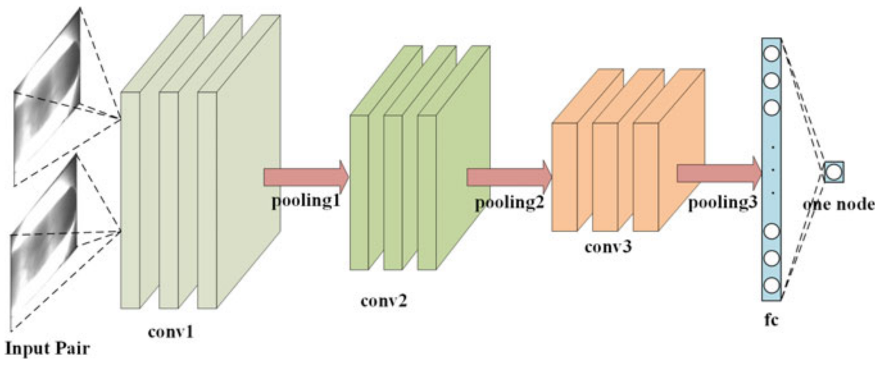

Fig. 5.8 The architecture for Modified VGG-16 for finger vein image matching

Table 5.4 Details of layer information of Modified VGG-16

Index	Type	Filter size	Num	Stride	Pad
1	conv1a	3	64	1	1
2	ReLU1a	–	–	–	–
3	conv1b	3	64	1	1
4	ReLU1b	–	–	–	–
5	conv1c	3	64	1	1
6	ReLU1c	–	–	–	–
7	pooling1	2	–	2	0
8	conv2a	3	128	1	1
9	ReLU2a	–	–	–	–
10	conv2b	3	128	1	1
11	ReLU2b	–	–	–	–
12	conv2c	3	128	1	1
13	ReLU2c	–	–	–	–
14	pooling2	2	–	2	0
15	conv3a	3	256	1	1
16	ReLU3a	–	–	–	–
17	conv3b	3	256	1	1
18	ReLU3b	–	–	–	–
19	conv3c	3	256	1	1
20	ReLU3c	–	–	–	–
21	pooling3	2	–	2	0
22	fc1	–	512	–	–
23	Dropout	–	–	–	–
24	fc2	–	1	–	–

the intra-class variation, and we suppose that $\boldsymbol{\mu}$ and $\boldsymbol{\varepsilon}$ are two independent variables with Gaussian distribution $N(0, \boldsymbol{\sigma}_{\mu})$ and $N(0, \boldsymbol{\sigma}_{\varepsilon})$.

Let I be the hypothesis that $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are from the same class, and E mean that they are from different class. Thus we could write the goal as to enlarge $\frac{P(f_1 f_2 | I)}{P(f_1 f_2 | E)}$ where $P(\cdot)$ is the distribution. For simplicity, we use log formulation $r(\boldsymbol{f}_1, \boldsymbol{f}_2) = \log \frac{P(f_1 f_2 | I)}{P(f_1 f_2 | E)}$ and the computations are as detailed in [1].

Our objective has been to enlarge $r(\boldsymbol{f}_1, \boldsymbol{f}_2)$ and therefore we compute maximum of results rather than the minimum while using the L2-norm. The CNN training part is the same as LCNN. After extraction of features, we retain 80 dimensions instead of original 500 to accelerate computations and use these features to compute matrices for the joint Bayesian formulation based classification.

5.4 Supervised Discrete Hashing

One of the key challenges for the successful usage of biometrics technologies are related to efficient search speed (fast retrieval) and template storage/size. Hashing is one of the most effective approaches to address such challenges and can *efficiently* encode the biometrics templates using binary numbers (2000 in our experiments) that closely reflect similarity with the input data/templates. With such strategy we can only store the corresponding short/compact binary codes, instead of original feature templates, and significantly improve the search or the matching speed by highly efficient pairwise comparisons using the Hamming distance.

This framework for an effective supervised hashing scheme is introduced in [15] and the objective in the learning phase is to generate binary codes for the linear classification. We firstly define the problem and assume that we have n samples/features $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_n]$ and our goal is to recover corresponding binary codes $\mathbf{B} = [\mathbf{b}_1 \mathbf{b}_2 \ldots \mathbf{b}_n]$ where $\mathbf{b}_i \in \{-1, 1\}$, $i = 1, 2, \ldots, n$. Since we have labels, in order to make good use of these information, we define a multi-class classification function:

$$\hat{\mathbf{y}} = \mathbf{W}^{\mathrm{T}} \mathbf{b} \text{ where } \mathbf{W} = [\mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_C], \tag{5.5}$$

where C is the total number of classes, and $\hat{\mathbf{y}} \in \mathbb{R}^{C \times 1}$ is the label vector, where the maximum one indicates its class of input $\mathbf{x}$. Now we can formulate the hashing problem as follows:

$$\min_{\boldsymbol{B}, \boldsymbol{W}, F} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}} \mathbf{b}_i) + \lambda \|\mathbf{W}\|^2, \text{ s.t. } \mathbf{b}_i = \operatorname{sgn}(F(\mathbf{x}_i)) \tag{5.6}$$

where $L(\cdot)$ represents the loss function used by us which is the L2-norm in our experiments, λ is the regularization parameter, and at the same time, $\mathbf{b}_i$ is generated by the hash function $\operatorname{sgn}(F(\mathbf{x}_i))$ where $\operatorname{sgn}(\cdot)$ is the sign function. With the help of Lagrange Multiplier, we can then rewrite (5.6) as:

$$\min_{\boldsymbol{B}, \boldsymbol{W}, F} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}}\mathbf{b}_i) + \lambda\|\mathbf{W}\|^2 + \mu \sum_{i=1}^{n} \|\mathbf{b}_i - F(\mathbf{x}_i)\|^2 \tag{5.7}$$

where μ is the Lagrange multiplier. We further select a nonlinear form of function for $F(\mathbf{x})$:

$$F(\mathbf{x} = \mathbf{U}^{\mathrm{T}}\phi(\mathbf{x}) \tag{5.8}$$

where $\mathbf{U}$ is the parameter matrix and $\phi(\mathbf{x})$ is a k-dimensional kernel that

$$\phi(\mathbf{x}) = \begin{bmatrix} \exp(\|\frac{\mathbf{x}-a_1\|^2}{\sigma}) \\ \ldots \\ \exp(\|\frac{\mathbf{x}-a_k\|^2}{\sigma}) \end{bmatrix} \tag{5.9}$$

$\boldsymbol{a}_j, j = 1, 2, \ldots, k$ are randomly selected anchor vectors from input. In order to compute $\boldsymbol{U}$ in the function, we can rewrite (5.6) as follows:

$$\min_{\mathbf{U}} \sum_{i=1}^{n} \|\mathbf{b}_i - F(\mathbf{x}_i)\|^2 = \min_{\mathbf{U}} \|\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\|^2 \tag{5.10}$$

where $\Phi(\mathbf{X}) = \{\phi(\mathbf{x}_i)\}_{i=1}^{n}$ and our purpose is to set the gradient to zero, which is

$$\nabla_{\mathbf{U}}\left(\|\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\|^2\right) = 2\left(\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\right)\Phi(\mathbf{X})^{\mathrm{T}} = 0 \tag{5.11}$$

It is simpler to achieve the final computation for $\mathbf{U}$ as follows:

$$\mathbf{U} = (\Phi(\mathbf{X})\,\Phi(\mathbf{X})^{\mathrm{T}})^{-1}\Phi(\mathbf{X})\,\mathbf{B}^{\mathrm{T}} \tag{5.12}$$

In order to solve for $\mathbf{W}$, we make use of the same method, first simplify (5.6) to

$$\min_{\mathbf{W}} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}}\mathbf{b}_i) + \lambda\|\mathbf{W}\|^2 = \min_{\mathbf{W}} \|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \lambda\|\mathbf{W}\|^2, \tag{5.13}$$

and then calculate its gradient based on $\mathbf{W}$

$$\nabla_{\mathbf{W}}(\|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \lambda\|\mathbf{W}\|^2) = 2\mathbf{B}\left(\mathbf{B}^{\mathrm{T}}\mathbf{W} - \mathbf{Y}^{\mathrm{T}}\right) + 2\lambda\mathbf{W}, \tag{5.14}$$

which can be set as zero and we get

$$\mathbf{W} = \left(\mathbf{B}\mathbf{B}^{\mathrm{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{B}\mathbf{Y}^{\mathrm{T}}, \tag{5.15}$$

Finally, we can solve for $\mathbf{B}$ and we exclude those variables which have no relation to $\mathbf{B}$ and then rewrite (5.6) as follows.

$$\min_{\mathbf{B}} \|\mathbf{Y}-\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2+\mu\|\mathbf{B}-F(\mathbf{X})\|^2=\min_{\mathbf{B}} \|\mathbf{Y}\|^2-2\mathrm{tr}\left(\mathbf{Y}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{B}\right)$$
$$+\|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2+\mu(\|\mathbf{B}\|^2+2\mathrm{tr}\left(\mathbf{B}^{\mathrm{T}}F(\mathbf{X})\right)+\|F(\mathbf{X})\|^2) \tag{5.16}$$

or we can further simplify the above formulation as follows:

$$\min_{\mathbf{B}} \|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2-2\mathrm{tr}\left(\mathbf{B}^{\mathrm{T}}(F(\mathbf{X})+\mathbf{W}\mathbf{Y})\right) \tag{5.17}$$

$\|\mathbf{B}\|^2$ is excluded here because $\mathbf{b}_i \in \{-1,1\}, i=1,2,\ldots,n$, indicating that $\|\mathbf{B}\|^2$ is some constant.

We can now solve this problem *bit-by-bit*. Let $\mathbf{p}^{\mathrm{T}}$ represent the lth row of $\mathbf{B}$, and $\mathbf{B}'$ is the matrix without $\mathbf{p}^{\mathrm{T}}$. Similarly let $\mathbf{v}^{\mathrm{T}}$ be the lth row of $\mathbf{W}$ and let $\mathbf{q}^{\mathrm{T}}$ be the lth row of $\mathbf{Q}$ where $\mathbf{Q}=F(\mathbf{X})+\mathbf{W}\mathbf{Y}$ then we can ignore $\mathbf{W}'$ and $\mathbf{Q}'$ While moving a row to the end for all matrices would not cause problems but help to better understand the problem. In order to enhance clarity of the problem, we can move all the lth row to the end and rewrite $\mathbf{B}=\begin{bmatrix}\mathbf{B}'\\ \mathbf{p}^{\mathrm{T}}\end{bmatrix}$, and the same for $\mathbf{W}$ and $\mathbf{Q}$ We can then rewrite first term in (5.17) as follows.

$$\|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2=\left\|\begin{bmatrix}\mathbf{W}'^{\mathrm{T}} & \mathbf{v}\end{bmatrix}\begin{bmatrix}\mathbf{B}'\\ \mathbf{p}^{\mathrm{T}}\end{bmatrix}\right\|^2=\|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2+\|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2+2\mathrm{tr}\left(\mathbf{B}'^{\mathrm{T}}\mathbf{W}'\mathbf{v}\mathbf{p}^{\mathrm{T}}\right)$$
$$=\|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2+\mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right)+2\left(\mathbf{W}'\mathbf{v}\right)^{\mathrm{T}}\mathbf{B}'\mathbf{p} \tag{5.18}$$

While $\|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2+\mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right)$ is equal to some constant, and because our goal is to solve for $\mathbf{p}$ we can regard other parts as constant. Here $\|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2$ is ignored because $\|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2=\|\mathbf{p}\mathbf{v}^{\mathrm{T}}\|^2=\mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right)=n\,\mathrm{tr}\left(\mathbf{v}\mathbf{v}^{\mathrm{T}}\right)$ where $\mathbf{p}^{\mathrm{T}}\mathbf{p}=n$. The other part can be simplified as follows:

$$\mathrm{tr}\left(\mathbf{B}^{\mathbf{T}}\mathbf{Q}\right)=\mathrm{tr}\left(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}'+\mathbf{p}\mathbf{q}^{\mathrm{T}}\right)=\mathrm{tr}\left(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}')+\mathrm{tr}(\mathbf{p}\mathbf{q}^{\mathrm{T}}\right)=\mathrm{tr}(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}')+\mathbf{q}^{T}\boldsymbol{p} \tag{5.19}$$

Combining these terms, we can rewrite (5.17) as follows.

$$\min_{\mathbf{B}} \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\mathbf{p}-\mathbf{q}^{\mathrm{T}}\mathbf{p}=\min_{\mathbf{B}}(\mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}'-\mathbf{q}^{\mathrm{T}})\mathbf{p} \tag{5.20}$$

This is an optimization problem, and $\mathbf{p}\in\{-1,1\}^n$, therefore we just need to incorporate the opposite sign for its first argument.

$$\mathbf{p} = -\text{sgn}\left(\mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}' - \mathbf{q}^{\mathrm{T}}\right) = \text{sgn}(\mathbf{q}^{\mathrm{T}} - \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}') \tag{5.21}$$

We can now *explicitly* outline computed parts in the following.

$$\mathbf{W} = \left(\mathbf{B}\mathbf{B}^{\mathrm{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{B}\mathbf{Y}^{\mathrm{T}}, \tag{5.15}$$

$$\mathbf{U} = (\Phi\left(\mathbf{X}\right)\Phi\left(\mathbf{X}\right)^{\mathrm{T}})^{-1}\Phi\left(\mathbf{X}\right)\mathbf{B}^{\mathrm{T}} \tag{5.12}$$

$$\mathbf{p} = \text{sgn}(\mathbf{q}^{\mathrm{T}} - \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}') \tag{5.21}$$

Shen et al. [15] have provided another computation based on the hinge loss. However for the simplicity, we incorporated L2-norm in our experiments and therefore this part is excluded here. We now have the required equations here and can *summarize* this algorithm as follows.

```
Algorithm: Supervised Discrete Hashing

Input: Training data {$\mathbf{X}, \mathbf{Y}$}
Output: Binary codes $\mathbf{B}$

1. Randomly select k anchors $\boldsymbol{a}_j$, $j = 1, 2, \ldots, k$ from $\mathbf{X}$ and calculate $\Phi\left(\mathbf{X}\right)$
2. Randomly initiate $\mathbf{B}$
3. Loop until converge or reach maximum iterations
    – Calculate $\mathbf{W}$ and $\mathbf{U}$ which are described in (5.15) and (5.12)
    – Learn $\mathbf{B}$ bit by bit, with the help of (5.21)
```

5.5 Experiments and Results

This section provides details on the experiments performed using various CNN architectures discussed in previous sections.

5.5.1 Database and Evaluation Protocol

In order to ensure reproducibility of experimental results, we utilized publicly available *two-session* finger vein images database from [17]. This database of 6264 images has been acquired from 156 different subjects and includes finger vein images from two fingers for each subject. However, second session images are only from 105 different subjects. This database has high intra-class variations in the images and includes significant variations in the quality of images which makes it most suitable for benchmarking the finger vein matching algorithms for the real applications. In our experiments, we only used first session images to train different network architectures discussed in previous section, and initially excluded 51 subjects without second session images. The experimental results are presented using independent second session test data. Therefore, each of the receiver operating characteristics uses 1260 (210 × 6) genuine scores and 263340 (210 × 209 × 6) impostor scores.

We experimented on ROI images, enhanced ROI images, and even Gabor- filtered images separately. The ROI images have 256 × 513 pixels, enhanced ROI images have 218 × 488 pixels, and even Gabor filtered images have 218 × 488 pixels. The experimental results using ROC and CMC from the respective CNN architecture are presented in the following.

5.5.2 Results Using LCNN

We first performed the experiments using the LCNN trained using the ROI images, enhanced ROI images, and the enhanced images using even Gabor filters (Figs. 5.2, 5.3, and 5.4). The experimental results using respective second session dataset are shown in Fig. 5.9. The respective ROC and CMC illustrate that *enhanced* ROI images can achieve superior matching performance than those from using ROI images. The enhanced ROI images using even Gabor filters significantly helps to *suppress* the noisy pixels and accentuate the vascular regions and is the plausible reason for superior accuracy.

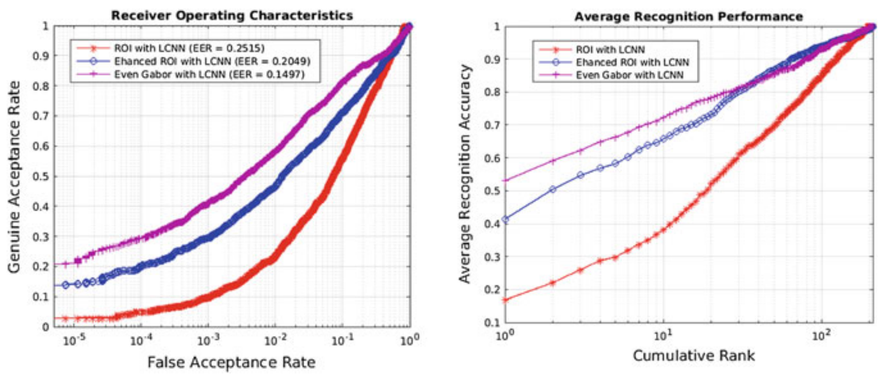

Fig. 5.9 Comparative ROC (*left*) and CMC (*right*) performance using LCNN

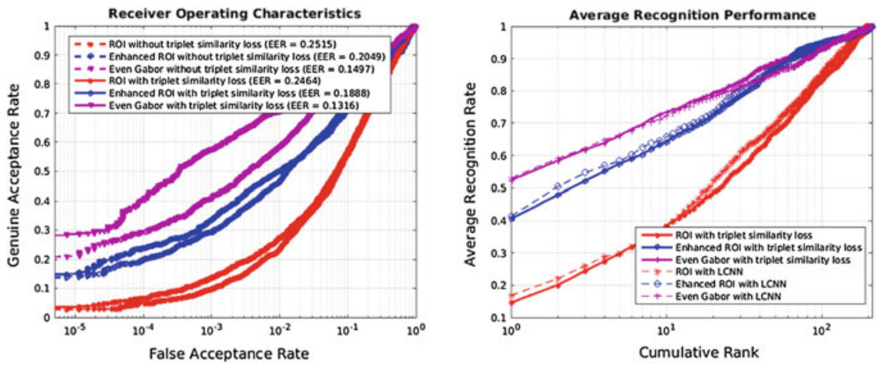

Fig. 5.10 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with triplet similarity loss

5.5.3 *Results Using LCNN with Triplet Similarity Loss Function*

The experimental results using LCNN trained with Siamese triplet similarity loss function are presented in Fig. 5.10. These results consistently illustrate superior performance using the architecture than the LCNN. The performance from the ROC of enhanced ROI with Gabor filters is *significantly* superior and this observation is in line with the trend observed from results using LCNN in Fig. 5.9 (the dash lines in Fig. 5.10 are previous results using LCNN for ease in the comparison). However, this approach has little influence on CMC.

The LCNN *without* triplet similarity loss tries to match a sample with its label, while LCNN *with* similarity loss focuses on the similarities of the images which could contribute to the better ROC performance. However, at the same time, label information is not sufficiently exploited with the triplet similarity loss and therefore the CMC performance has not changed significantly.

5.5.4 *Results Using CNN and Joint Bayesian Formulation*

Another scheme that has shown superior performance for ocular classification in [23] uses joint Bayesian [1] instead of L2-norm as the metrics for the similarity. The LCNN with the joint Bayesian classification scheme was also attempted to ascertain the performance. The ROC using this approach is illustrated in Fig. 5.11 where dash lines are previous results using LCNN and indicates performance improvement over LCNN.

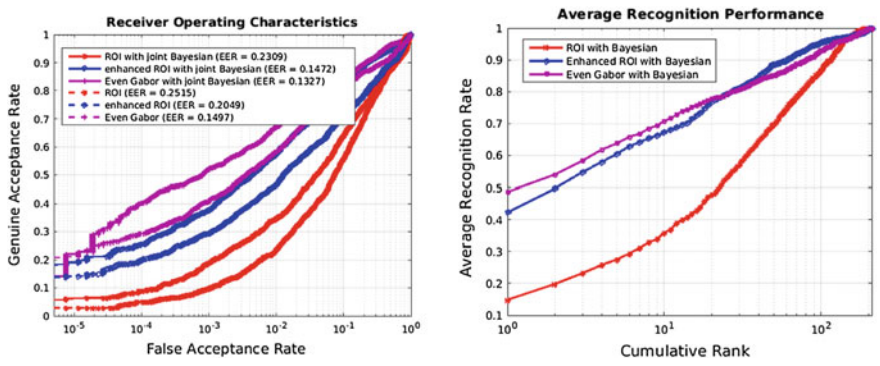

Fig. 5.11 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with Joint Bayesian

5.5.5 Comparisons and Results Using Supervised Discrete Hashing

The supervised discrete hashing (SDH) scheme detailed in Sect. 5.4 was also investigated for the performance improvement. Only *first* session data was employed for the training part and the used for generating binarized bits that were used for matching using Hamming distance. The results using the ROC and CMC in Fig. 5.12 illustrates *consistent* performance improvement with the usage of SDH and the trends in the usage of enhanced ROI images are also consistent with our earlier observations.

The LCNN trained with triplet similarity loss function was also employed and used with the SDH to evaluate the performance. We attempted to ascertain the performance with different number of bits. Higher number of bits for SDH can be generally expected to offer superior results, but requires more training time. It should be noted that this method is actually a second-step training, and tries to map features from

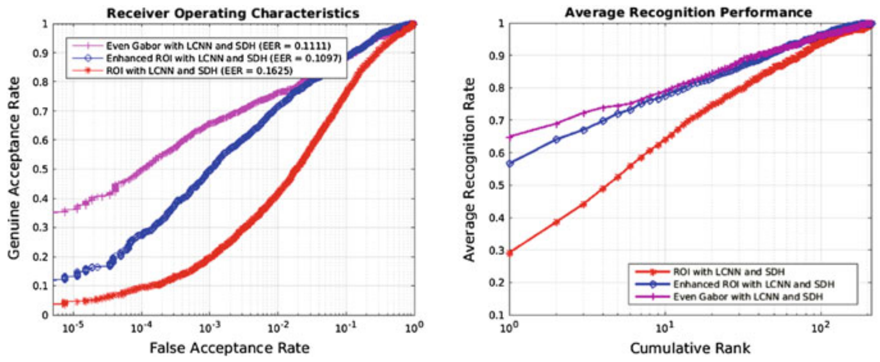

Fig. 5.12 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with SDH

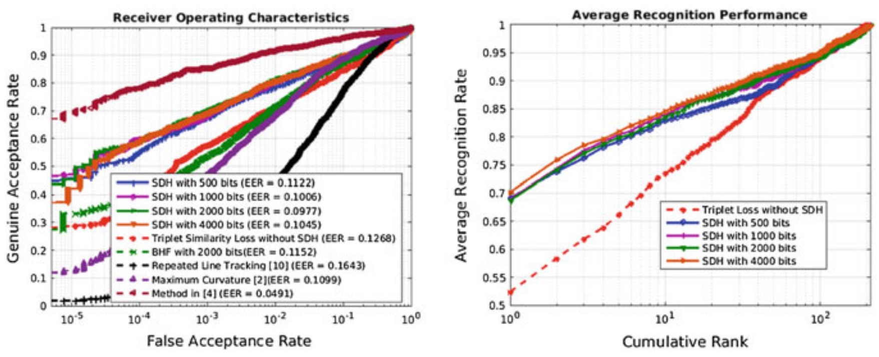

Fig. 5.13 Comparative ROC (*left*) and CMC (*right*) performance using SDH with triplet similarity loss function

the Euclidian space to the binary space. The training phase and hamming distance metrics can contribute to its superior performance. The combination of CNN with the hashing to reduce for the faster and real-time identification has also been attempted earlier [18] but for the face recognition problem. Authors in [18] introduced incorporated Boosted Hashing Forest (BHF) for the hashing and therefore we also attempted to incorporate BHF scheme to comparatively evaluate the performance. However, our results illustrated superiority of SDH over BHF and the template size using BHF was also fixed to 2000 bits. Although our results did not achieve significant improvement in the performance using BHF, its usage helps in remarkably reducing the template size. In order to ascertain comparative performance for matching finger vein images using the handcrafted features, we also performed additional experiments. The ROC from the same test images and matching protocols but using repeated line tracking [1] (also used as baseline in [21]) and curvatures [13] method is illustrated in Fig. 5.13. We can observe that the experimental results using SDH and LCNN offer superior performance and significantly reduced template size. Our results over the method using [8] are can be considered as competing and not yet superior but offers significantly reduced template size (∼26 times smaller) over the best of the methods in [8] which is still the state-of-the-art method to date.

5.5.6 Results Using Modified VGG-16

The experimental results using modified VGG-16, as detailed in Sect. 5.3.3 are presented in Fig. 5.14. It should be noted that this CNN architecture generates single match score and therefore we cannot use SDH scheme to the infer features. We can infer from the ROCs in Fig. 5.14, that modified VGG-16 architecture generates superior performance for matching finger vein images as compared to the network trained using LCNN. This architecture directly generates the matching scores and

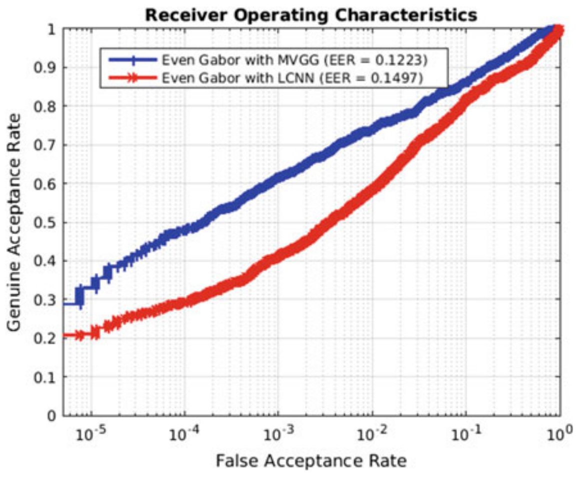

Fig. 5.14 Comparative performance using SDH with MVGG

therefore can minimize the problems from the inappropriate choice of distance metric (or weighting of features), which may be a convincible reason for its superior performance/results. It should however be noted that different training of the network can lead to variations in the results.

5.5.7 Results Using Single Fingers and Unregistered Fingers

Since we used both finger vein images from two fingers to form larger dataset (as in [8]) for above experiments, it is judicious to ascertain the performance when only one, i.e., index or middle, finger vein images are used in the training and test phase. The results using respective finger vein images from 105 different subjects are comparatively shown in Fig. 5.15 using the ROC. The performance using the images from both fingers (using 210 class formed by combination of index and middle finger for 105 different subjects) is superior to single finger, and index finger shows better performance than middle finger. Similar trends are also observed in [8] and can be attributed to the nature of dataset.

In earlier experiments, the first session data had images acquired from the same subjects who were providing their images during the second session and were used as test set for the performance. In order to ascertain robustness of self-learned features using the best scheme so far, we also evaluated the performance from the 51 subjects in this dataset which did not have any two-session finger vein images Therefore, images from none of these subject's images were employed during the training for CNN in any of the earlier experiments. The six images from these 51 subjects were used to ascertain performance using challenging protocol, i.e., all-to-all, so that we generated a total of 1530 genuine scores and 185436 impostor scores to ascertain such performance. The ROC corresponding to this independent test subjects finger

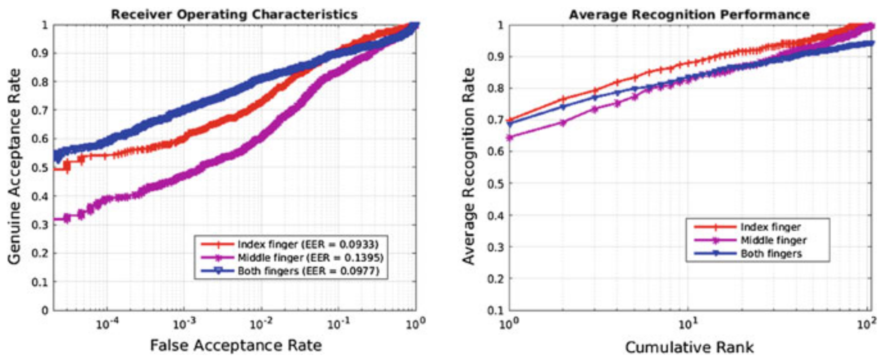

Fig. 5.15 Comparative ROC (*left*) and CMC (*right*) performance using SDH with triplet similarity loss experimented on single finger

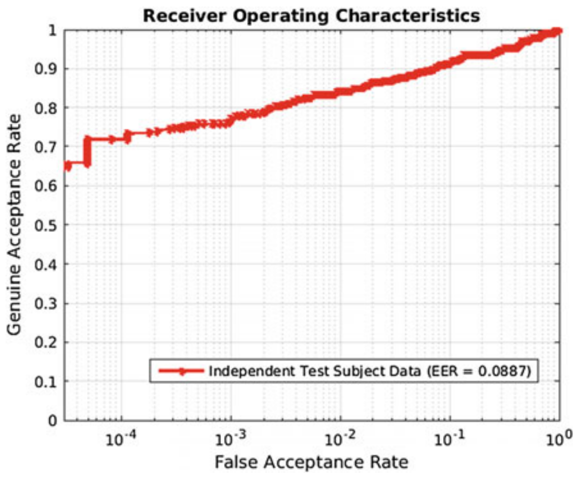

Fig. 5.16 The performance using independent test subjects in [17] for matching finger vein images

vein data is shown in Fig. 5.16 and the results indicate promising performance from the self-learned features using a model trained (in Sect. 5.3.2 and SDH) for matching finger vein images from unknown subjects.

5.6 Discussion

This chapter has investigated finger vein matching performance using various convolutional neural network architectures. Unlike earlier work on finger vein image matching which largely employed handcrafted features, our emphasis has been to investigate automatically learned features using the capabilities of deep learning. We systematically investigated the performance improvement using just the ROI images and the systematically enhanced images that mainly emphasizes on subsurface vascular network. Our results consistently indicate superior performance from the CNN that are trained with images which have such enhanced vascular features.

According to our experimental results in Sect. 5.5.6, modified VGG-16 (MVGG) achieves superior performance than LCNN. However, MVGG requires significantly higher time for the training (also for the test phase). This can be largely attributed to the fact that it directly generates the match scores and therefore the loss function outputs propagate iteratively through the whole network to ascertain the similarity between a pair of finger vein images. At the same time, we cannot incorporate SDH (hashing scheme) with the MVGG, due to nonavailability of intermediate features, while the usage of SDH has shown to offer remarkable improvement in the performance.

It should be noted that the triplet similarity loss function helps to significantly improve the experimental performance using the LCNN. However, this approach cannot adequately make use of the label information, because it attempts to decrease the feature similarity between the pairwise images from the same subject, but cannot accurately locate the labels, i.e., identity of the subjects they are associated with. Supervised discrete hashing approach significantly improves performance and the retrieval speed, and decrease the storage which requires only 250 bytes (2000 bits) for the one template (feature vector). However, it should also be noted that this method needs a separate training phase and training time rapidly increases when the bit length or number of features are increased.

The work detailed in this chapter also had several constraints and therefore should be considered only preliminary. The database employed, although one of the largest two-session finger vein databases available in public domain, is still of smaller size for the deep learning based algorithms. There are several references in the literature that have shown promising performance but yet to demonstrate superior matching performance over the method in [8] using fair comparison or the same matching protocols. Therefore, we are justified in using the performance from [8], for this publicly available dataset, as the reference.

5.7 Conclusions and Further Work

This chapter has systematically investigated finger vein identification using the various CNN architectures. Unlike earlier work on finger vein matching which employed handcrafted features, our emphasis has been to investigate performance from the automatically learned features using the capabilities of deep learning. We systematically investigated the performance improvement using the ROI finger vein images, and the enhances images and consistently observed that the usage of ROI images with enhanced vascular features and attenuation of background (noise) can significantly improve the performance. The factors that most influence the accuracy of matching finger vein images is the depth of the network, the pretraining and the data augmentation in terms of random crops and rotations.

The LCNN architecture detailed in Sect. 5.3.2 that uses triplet similarity loss function achieves superior performance than those from the original *softmax* loss. Our experimental results using SDH illustrates that this hashing method significantly improves the matching performance and offers better alternative than BHF for the finger vein identification problem. Apart from two-session identification on same subjects, we also experimented on the datasets from subjects whose data was never available for training the CNN and this remarkable performance indicates high generalization capability for the finger vein identification problem. Our proposal for finger vein identification detailed in this chapter achieves smallest template size than using any other methods available in the literature to date. This work however had several constraints and therefore should be considered only preliminary. The database employed, although the largest two session finger vein images database available in public domain, is still of smaller size for the deep learning algorithms. Despite promising improvements in the accuracy from the publicly available (limited) training data, more work needs to be done to achieve significantly superior results than using the best performing method in [8]. Further work should use larger training dataset but should provide performance using the independent test data or using the publicly available dataset [17] to achieve more accurate alternative for the automated finger vein identification.

References

1. D. Chen, X. Cao, L. Wang, F. Wen, J. Sun, Bayesian face revisited: a joint formulation, in *Proceedings of ECCV 2012* (2012)
2. L. Chen, J. Wang, S. Yang, H. He, A finger vein image-based personal identification system with self-adaptive illuminance control. IEEE Trans. Instrum. Meas. **66**(2), 294–304 (2017)
3. L. Dong, G. Yang, Y. Yin, F. Liu, X. Xi, Finger vein verification based on a personalized best patches map, in *Proceedings of 2nd IJCB 2014*, Tampa, Sep.–Oct. 2014
4. F. Hillerstorm, A. Kumar, R. Veldhuis, Generating and analyzing synthetic finger-vein images, in *Proceedings of BIOSIG 2014*, Darmstadt, Germany, September 2014
5. https://polyuit-my.sharepoint.com/personal/csajaykr_polyu_edu_hk/_layouts/15/guestaccess.aspx?docid=11d706337994749759a2cc64cb70b604a&authkey=AcDq15geZ7942-7owNQfmYQ
6. M. Kono, H. Ueki, S. Umemura, A new method for the identification of individuals by using of vein pattern matching of a finger, in *Proceedings of 5th Symposium on Pattern Measurement*, pp. 9–12 (*in Japanese*), Yamaguchi, Japan, 2000
7. M. Kono, H. Ueki, S. Umemura, Near-infrared finger vein patterns for personal identification. Appl. Opt. **41**(35), 7429–7436 (2002)
8. A. Kumar, Y. Zhou, Human identification using finger images. IEEE Trans. Image Process. **21**, 2228–2244 (2012)
9. C. Liu, Y.H. Kim, An efficient finger-vein extraction algorithm based on random forest regression with efficient local binary patterns, in *2016 IEEE ICIP* (2016)
10. Y. Lu, S. Yoon, D.S. Park, Finger vein identification system using two cameras. Electron. Lett. **50**(22), 1591–1593 (2014)
11. Y. Lu, S. Yoon, S.J. Xie, J. Yang, Z. Wang, D.S. Park, Efficient descriptor of histogram of salient edge orientation map for finger vein recognition. Optic. Soc. Am. **53**(20), 4585–4593 (2014)

12. N. Miura, A. Nagasaka, T. Miyatake, Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification, in *Machine Vision & Applications*, pp. 194–203, Jul, 2004
13. N. Miura, A. Nagasaka, T. Miyatake, Extraction of finger-vein patterns using maximum curvature points in image profiles, in *Proceedings of IAPR Conference on Machine Vision and Applications*, pp. 347–350, Tsukuba Science City, May 2005
14. F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering (2015), arXiv:1503.03832
15. F. Shen, C. Shen, W. Liu, H.T. Shen, Supervised Discrete Hashing, in *Proceedings of CVPR, 2015*, pp. 37-45, Boston, June 2015
16. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556
17. The Hong Kong Polytechnic University Finger Image Database (Version 1.0) (2012), http://www.comp.polyu.edu.hk/~csajaykr/fvdatavase.htm
18. Y. Vizilter, V. Gorbatsevich, A. Vorotnikov, N. Kostromov, Real-time face identification via CNN and boosted hashing forest, in *Proceedings of CVPR 2016 Biometrics Workshop, CVPR'W 2016*, Las Vegas, June 2016
19. C. Xie and A. Kumar, 'Finger vein identification using convolutional neural networks', The Hong Kong Polytechnic University, Tech. Report No. COMP K-25, Jan 2017
20. X. Wu, R.He, Z. Sun, T. Tan, A light CNN for deep face representation with noisy labels (2016), arXiv:1151.02683v2
21. Y. Ye, L. Ni, H. Zheng, S. Liu, Y. Zhu, D. Zhang, W. Xiang, FVRC2016: the 2nd finger vein recognition competition, in *2016 ICB* (2016)
22. Y. Zhou, A. Kumar, Contactless palmvein identification using multiple representations, in *Proceedings of BTAS 2010*, Washington DC, USA, September 2010
23. Z. Zhao, A. Kumar, Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network. IEEE Trans. Info. Forensics Secur. **12**(5), 1017–1030 (2017)