Deep Convolutional neural network for Fingerprint Pattern Classification

Anonymous IJCB 2017 submission

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous BTAS/IJCB abstracts to get a feel for style and length.

1. Introduction

Fingerprints are ridge and valley patterns presented on the surface of human fingertips. Fingerprints are used to recognize humans for applications such as verifying an identity claim (*i.e.*, one-to-one search to unlock a smartphone, for example), or identification (*i.e.*, one-to-many search to find a suspect of a crime, for instance). Typically, to query a fingerprint, the system needs to search and compare the query print with the fingerprints stored in a reference (or enrolled) database. The size of a reference database can be from thousands to hundreds of millions subjects, depending on the application. For example, the Aaddhar project in India has enrolled 111,98,29,743 persons as of February 18, 2017 [2]. As the size of the database grows, the number of comparisons to be made for identification purposes grow, so does the computation time. To mitigate this problem, most fingerprint recognition algorithms first classify a fingerprint into a basic pattern type and then perform fingerprint matching within fingerprints of that type. The major five fingerprint pattern types used today are an extension of the three pattern types (whorl, loop, and Arch) introduced by Henry Faulds (Henry classification system [12]) and Sir Francis Galton [10] in late 19th century. These five pattern types are: arch, left Loop, right Loop, tented arch and whorl, see Fig.1. Because arch and tented arch only accounts for a small portion (around 6%) in human, some automatic fingerprint identification systems combine these two classes into one class.

As mentioned above, to manage the computation load, large scale fingerprint identification algorithms employ multi-stage matching whose first step is often filtering based on fingerprint pattern type. As such the accuracy of the fingerprint classification algorithm largely influences the identification accuracy. An error in finger pattern classification will propagate throughout the system, and ultimately result in an recognition error. In this project (paper) we propose an automated fingerprint pattern classification that is not based on feature extraction.

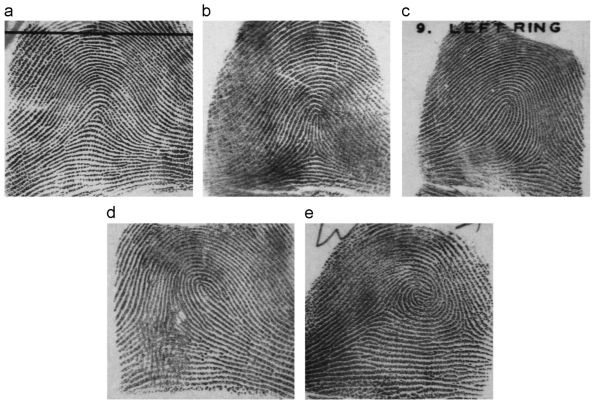


Figure 1: Examples of fingerprint classes[6]: (a) Arch (b) Tented Arch (c) Left Loop (d) Right Loop (e) Whorl.Note that tented and tented arch are similar.

2. Problem statement

The challenge of classifying fingerprint includes: 1) quality of fingerprints, particularly poor quality; 2) small inter-class dissimilarity and small intra-class similarity; for example, tented arch and loop may look similar; 3) ambiguities in some labels (pattern class; some fingerprints can be classified into multiple classes, or different classes by different fingerprint experts.

Previous work mostly consists of singularity points (core and delta) detection or extracting features such as ridge and orientation flow, or using human markups (or handcrafted features) as the basis for pattern type classification. Therefore, the accuracy of these methods depends on the goodness (or utility) of the selected features and the precision of

the feature extraction portion of the algorithms. Both are sensitive to the noise and the variations of the gray-scale level of the input image. Using handcrafted features can improve performance. However, in addition to being burdensome and time consuming, accuracy of handcrafted features cannot be guaranteed due to the existence of noise and poor image quality. Moreover, their repeatability and reproducibility cannot be guaranteed either, due to inter- and intra-examiners variations [20]. Our approach differs from these works in the sense that we aim to use raw images instead of features as input. Convolutional neural network (CNN) has the capability of learning features and it can be directly applied on raw images. CNN also exhibits powerful classification capability in many areas[17][19].

An overview of related work follows. Karu and Jain [14] presented a rule-based classifier based on extracting singular points. Fitz and Green [9] used a Hexagonal Fourier Transform to classify fingerprints into whorls, loops and arches. Jain *et al.* [13] used a bank of Gabor filters to compute a feature vector (FingerCode) and then used a K-nearest neighbor classifier and a set of neural networks to classify a feature vector into one of the five fingerprint pattern classes. Cappelli *et al.* [7] partitioned a fingerprint directional image into "homogeneous" connected regions according to the fingerprint topology, resulting in a synthetic representation which is then used as a basis for the classification. Bernard *et al.* [5] used the Kohonen topologic map for fingerprint pattern classification. Kai Cao *et al.*[6] proposed to extract fingerprint orientation feature and used a hierarchical classifier for classification. Ruxin Wang *et al.* [21] also used orientation filed as features. By adopting a stacked autoencoder, they achieved 93.1% in four-class classification.

3. Proposed research

In this project, we aim to develop and implement a deep learning algorithm that takes a fingerprint image as an input and classify it into one of the five pattern class types of a) Arch; b) Tented Arch; c) Left Loop; d) Right Loop; or e) Whorl.

3.1. Feature Extraction

We will first apply raw fingerprint images to train a CNN for classification. The outputs of some intermediate layer of CNN will be used as features for possibly a support vector machine classifier.

For CNN architecture, we will first use canonical architecture (such as 5 *convolutional* + 3 *fully-connected* in *AlexNet* [16]). We will then modify the CNN architecture to improve the performance.

3.2. Classifier

We will consider two classifiers. The first one is the prediction layer of CNN. The values in last layer indicates the predicted probabilities of each class. The second one is support vector machine (SVM) whose input features comprise of the CNNs middle or last layers.

3.3. Data Augmentation

To further improve the performance, we will use data augmentation technique to generate more training samples in order to increase the generalization ability of our model. The augmentation methods include image rotation, resizing and translation.

3.4. Multi-Task Learning

Multi-Task Learning (MTL)[8] aims to improve the performance of multiple classification tasks by learning them jointly. In MTL, some tasks can benefit from auxiliary information which is introduced by other tasks and the performance is improved[22]. In our project, the network is trained to perform both the primary task (fingerprint type classification) and one or more auxiliary tasks. The classification task is expected to benefit from the auxiliary task by jointly training them together. This process can also be viewed as incorporating human knowledge into to training procedure.

To obtain labels for auxiliary tasks, we will use existing methods. For example, orientation flow estimation in [1].

4. Dataset

In this project, we will use NIST Special Database 4 [4] for our experiments. Some samples can be seen in Fig.1. The NIST database of fingerprint images contains 2000 8-bit gray scale fingerprint image pairs, totally 4000 images. Each image is 512-by-512 pixels with 32 rows of white space at the bottom and classified using one of the five following classes: Arch, Left and Right Loops, Tented Arch, Whorl. Each of the five classes has 400 pairs. Each of the fingerprint pairs are two completely different rollings of the same fingerprint.

5. Methodology

The methodology scheme is shown in Figure.2. During training stage, the training images are preprocessed using data augmentation techniques to increase the data diversity. The augmented data are fed into deep convolutional neural network(ConvNet) for training. After deep ConvNet training is finished, it serves as a feature extractor and extracts deep features representing all training images. These deep features are used in the final step to train a support vector machine (SVM) for classification. During the testing stage,

the testing images are fed into the deep ConvNet for deep features extraction. These intermediate deep features are used as testing samples and fed into the trained SVM. The output of SVM is the final prediction.

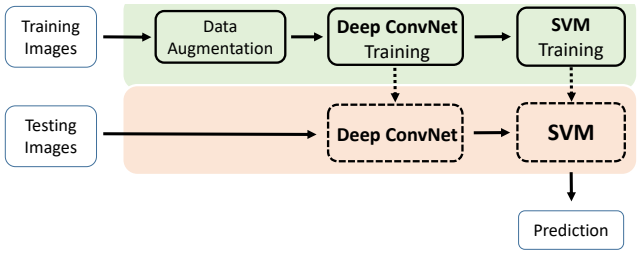


Figure 2: Overview of proposed approach.

5.1. Deep ConvNet Architecture

An overview of the network architecture is shown in Figure.4.

Our proposed network is based on deep residual network proposed by Kaiming He **et al.**[11]. Deep residual network has been proven to outperform other deep plain networks because it addresses the degradation problem by reformulating the layers as learning residual functions instead of learning unreferenced functions. Table1 shows the details of our network. The input size of our deep ConvNet is 512×512 and the number of channels is 1. The first two layers of our network are convolutional layers which has 96 7×7 filters and the stride is 2. The third layer has 64 7×7 filters and the stride is 2. *conv4*, *conv5*, *conv6* and *conv7* are composed of residual building blocks. Specifically, there is a max pooling layer before *conv4*. The parameters inside the brackets specify the residual building block size. The multiplier after bracket specifies the multiplicity of the that block in that layer. More details can be seen in [11]. The global pooling layer in *conv8* generates $1 \times 1, 2048$ output and the last layer uses 5 1×1 filters to generate the final prediction. The number of parameters of proposed deep ConvNet is 24.26 million. We use Relu[18] as intermediate activation function.

The novelty of our network is that we use 512×512 as network input size. The reason is that we preserve the detail of fingerprint as much as possible. Empirically we found the performance drop significantly when resizing the fingerprint samples to 224×224.

As shown in Figure3, after down-sampling the images to 224×224, the fingerprints are blurred, leading to the loss of important information such as deltas and cores. There are basically two solutions. The first one is to localize the fingerprint in the image and crop out a specific smaller region that mostly contains the fingerprint. This approach relies on the localization accuracy which maybe a bottleneck.

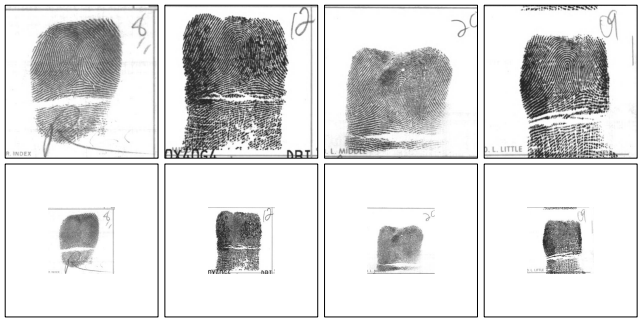


Figure 3: NIST SD14 Images under different resolutions. The top row are images in 512×512 and the bottom row are images in 224×224. Ridges of top images are more distinguishable than that of bottom images.

Another solution is to maintain the original solution of the fingerprint. However, this solution will result in huge computational cost and high memory usage when training. We adopt the second solution. After carefully looking at the fingerprint samples under different resolutions, we decide to use 512×512 as input size for two reasons. First, it is the original image size of NIST SD4. Second, resizing original NIST SD14 images to this resolution can preserve sufficient details for classification. To reduce the computational cost, we add *conv1* and *conv2* with stride 2 to down-sample the input images. As we can see in Table.1, after *conv2*, the feature map size is $128 \times 128, 96$. The spatial size is reduced (from512×512 to 128×128) and spatial information is stored in the increased channels (from 1 to 96).

5.2. Data Augmentation

Due to the wild nature of the fingerprints samples, the fingerprints may exhibit a wide range differences in terms of position, rotation, brightness and contrast. To enhance the generalization ability of our ConvNet, we adopt data augmentation techniques to increase the data diversity.

During training stage, the input images will go through a augmentation process. The data augmentation includes:

During training, we have applied three data augmentation techniques on segmented pill images to augment training dataset.

1. Random Cropping. The input images are first resized to 532×532. We randomly cropped a 512×512 region from the resized images.

2. Random Rotation. We randomly rotate the input images by ω degrees where $\omega \sim uniform(-30°, 30°)$.

3. Random Brightness. Random brightness change is performed on the input images. The gray scale of the

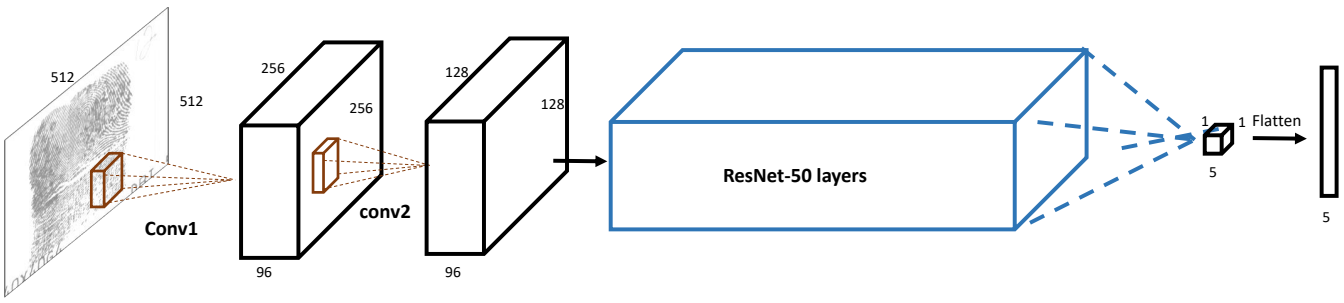


Figure 4: Architecture of proposed CNN.

Table 1: Detail of proposed deep ConvNet. The format is inspired by [11]

Layer	detail	Output size
conv1	$7 \times 7, 96$, stride=2	$256 \times 256, 96$
conv2	$7 \times 7, 96$, stride=2	$128 \times 128, 96$
conv3	$7 \times 7, 64$, stride=2	$64 \times 64, 64$
conv4_x	3×3 max pooling	$16 \times 16, 256$
	$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$	
conv5_x	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$	$8 \times 8, 256$
conv6_x	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 6$	$4 \times 4, 1024$
conv7_x	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$	$4 \times 4, 2048$
conv8	global pooling	$1 \times 1, 5$
	$1 \times 1, 5$	

input images I are change to $I + \delta$ where δ is sampled from $uniform(-50, 50)$.

4. Random Contrast. We randomly change the contrast of images. The contrast factor is sampled from $uniform(0.4, 1.6)$.

5.3. SVM

The deep ConvNet only serves as a feature extractor and we use non-linear SVM as a classifier. The kernel is radial basis function(RBF). The gamma of RBF kernel is set to be $\frac{1}{n}$ where n is the feature dimensionality. The penalty for error term C is set to be 1.0. We use the output of conv7_x as features. As such, each sample is represented by a feature vector $x \in \mathbb{R}^d$ where $d = 4 * 4 * 2048 = 32768$. The output of SVM is the predicted label $\hat{y}$ indicating one of the fingerprint classes.

6. Experiments

6.1. Dataset

In this project, we use NIST Special Database 4 [4] and NIST Special Database 14 [3] for our experiments. The NIST SD4 contains 2000 8-bit gray scale fingerprint image pairs, totally 4000 images. The size of each image is 512×512 and each image is classified using one of the five following classes: Arch, Left and Right Loops, Tented Arch, Whorl. Each of the five classes has 400 pairs(800 images). Each of the fingerprint pairs are two completely different rollings of the same fingerprint.

The NIST SD14 contains 27000 8-bit gray scale fingerprint image pairs. There are 2700 subjects in this dataset and each subject has 10 fingerprint samples pairs. The size of each image is 768×832. To fit in our network, we centrally crop 768×768 from the samples and resize them into 512×512. The distribution of fingerprint classes is as shown in Table.2. We can see that unlike NIST SD4, Arch and Tented Arch samples are only a small portion of the NIST SD14.

Table 2: Class Distribution of NIST SD14.

Arch	Left Loop	Right Loop	Tented Arch	Whorl
3.6%	31.9%	30.5%	3.2%	30.8%

6.2. Experimental Setup

We use a i7-5930K desktop with 32GB memory and a Nvidia GTX TITAN X GPU for experiments. Typically, we use Tensorflow 1.0.1 as the deep learning library and Adaptive Moment Estimation(Adam[15]) as the optimization algorithm. The learning rate is 0.0001. We also use ℓ_2 regularization with 0.0001 weight decay rate. The batch size is 32. We evaluate our approach on NIST SD4 and

NIST SD14 respectively. Each experiment is trained for $20k$ steps.

For NIST SD14 experiments, we use the samples of 80% subjects for training, totally 2160 subjects with 43200 images. Among these 432000 images, 36 of them have labels that do not belong to the 5 classes. These 36 images are discarded. The remained data of 20% subjects are used for testing, totally 10800 images. 9 of these images are discarded due to the same reason above.

For NIST SD4 experiments, we adopt two evaluation protocols. The first protocol is cross-sample for fair comparison with other works, where we use all the first samples in each fingerprint pair as training set and all the second samples as testing set. The second protocol is cross-finger, where we use 50% fingers for training and 50% for testing to ensure the same finger does not existing in training and testing set at the same time. To improve the performance for NIST SD4, We use NIST SD14 data to pre-train the model.

In addition to 5-class fingerprint classification, we also evaluate our SVM performance on 4-class fingerprint classification because 4 class classification are also used in other studies. To achieve 4-class classification, we merge Tented Arch class into Arch class when training SVM.

6.3. NIST SD14 result

The result for NIST SD14 is shown in Table3. In addition to report SVM performance, we also report the performance when ConvNet is used as classifier. As we can see, both ConvNet and SVM achieve the same accuracy (0.9861) for 5-class classification. However, ConvNet performs slightly better in terms of average precision, recall rate and F1 score. For 4-class classification, the 4-class SVM achieves 0.9875 accuracy.

For 5-class classification, the confusion matrix is shown in Figure.5a and Figure.5b. For 4-class classification, the confusion matrix is shown in Figure.6a. As we can see, the number of Arch and Tented Arch samples are relatively small compared to other classes. However, our proposed approach can still achieve high accuracy despite the unbalanced distribution of fingerprint types. Though computation based on Figure.5a, we can see that Tented Arch achieves the lowest precision (0.959) and recall rate(0.950) due to lack of training samples and label ambiguity. Based on Figure.6a, both the precision and recall rate of Arch increases to 0.98, indicating many mis-classified samples can be eliminated by combing Arch and Tented-Arch classes.

6.4. NIST SD4 Result

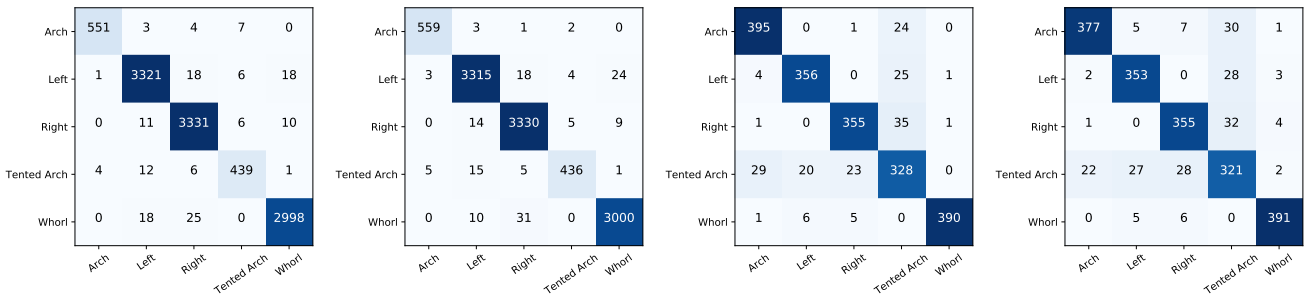
The result for NIST SD4 is shown in Table4. We have three observations from Table4. First, in both protocol, our proposed SVM performs better than ConvNet not only in accuracy but also in average precision, recall rate and F1 score. In cross-sample protocol, the accuracy of 5-class SVM is 0.9275, which is 0.006 higher than 5-class ConvNet. In cross-finger protocol, the accuracy of 5-class SVM is 0.912, 0.014 higher than 5-class ConvNet. Second, there is a performance drop in cross-finger compared to cross-sample. For 5-class ConvNet, the accuracy drops 0.023. For 5-class SVM, the accuracy drops 0.015. For 4-class SVM, the accuracy drops 0.011. SVM suffers the smaller performance drop than ConvNet if cross-finger protocol is used. The performance drop is small, indicating the generalization ability of our proposed method. Third, the accuracy is improved if Tented Arch and Arch are combined into one-class. The accuracy of 4-class SVM is 0.022 higher than 5-class SVM in cross-sample and 0.027 higher in cross-finger. This indicates many mis-classified samples can be eliminated by combing Arch and Tented-Arch classes, as in NIST SD 14.

Table 3: Experiment results for NIST SD14. In column 4, 5 and 6, we also report the average precision, recall and F1 score for all predicted classes.

method	# of classes	accuracy	average precision	average recall	average F1 score
ConvNet	5	0.9861	0.9843	0.9793	0.9817
SVM	5	0.9861	0.9822	0.9781	0.9801
SVM	4	0.9875	0.9869	0.9867	0.9868

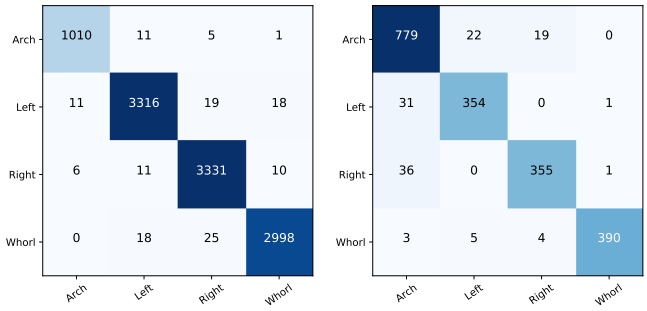
For 5-class classification using cross-finger protocol, the confusion matrices are shown in Figure.5c and Figure.5d. For 4-class classification using cross-finger protocol, the confusion matrices are shown in Figure.6b. We can see that both in Figure.5c and Figure.5d, the Tented Arch class achieves the lowest precision (91.8%) and lowest precision rate (94.0%) among five classes. In later experiments, we can see that the mis-labeled samples are ambiguous can be eliminated by introducing the second labels.

In NIST SD4, around 17% of the samples are ambiguous and are labeled with two classes. Many existing works report their best performance based on these additional labels. We also evaluate our approach using the additional 17% two labels to compare with other methods and the results are reported in Table.5. The training procedure remains the same where we only use one label for training. When testing, for those 17% samples, as long as the prediction for the test sample matches one of the two labels, the test sample is considered As we can see, a significant performance gain is obtained after the additional 17% are used. For cross-sample,our proposed ConvNet achieves 0.9535 accuracy, which is 0.032 higher than before. The proposed 5-class and 4-class SVMs achieve 1.0 accuracy and is the best among all the methods. For cross-finger, our proposed ConvNet achieves 0.945 accuracy, which is 0.046 higher than before. The proposed 5-class and 4-class SVMs still achieve 1.0 accuracy in this protocol.



(a) SVM for NIST SD14 (b) ConvNet for NIST SD14 (c) SVM for NIST SD4 (d) ConvNet for NIST SD4

Figure 5: Confusion Matrices for 5-class classification



(a) SVM for NIST SD14 (b) SVM for NIST SD4

Figure 6: Confusion Matrices for 4-class classification

Table 4: Experiment results for NIST SD4. In column 4, 5 and 6, we also report the average precision, recall and F1 score for all predicted classes.

method	# of classes	accuracy	average precision	average recall	average F1 score
Cross-Sample					
ConvNet	5	0.9215	0.9225	0.9215	0.9217
SVM	5	0.9275	0.9325	0.9275	0.9288
SVM	4	0.9495	0.9576	0.9459	0.9514
Cross-Finger					
ConvNet	5	0.8985	0.8991	0.8986	0.8987
SVM	5	0.9120	0.9132	0.9117	0.9123
SVM	4	0.9390	0.9452	0.9357	0.9403

6.5. Discussion

From experiment results we can see that our proposed approach can successfully perform fingerprint type classification on raw fingerprint images without the need of extracting hand-crafted features. Our proposed approach achieves the highest classification accuracy on both NIST SD14 and NIST 4 dataset to the best of our knowledge. Experiment results show that using Deep ConvNet as a feature extractor and train a SVM on top of the ConvNet can bring further performance gain compared to a standalone Deep ConvNet. Tented Arch and Arch fingerprints contributes the most error rate among the five classes. Many mis-classified labels can be corrected by combining Tented Arch and Arch classes into one class or using additional labels. We also collect some mis-classification cases on NIST SD14 and show them in Figure7. As we can see, in

Table 5: Experiment results for NIST SD4 with two labels.

method	# of classes	accuracy	protocol
ConvNet	5	0.9535	cross-sample
SVM	5	1.0	cross-sample
SVM	4	1.0	cross-sample
[6]	5	0.959	cross-sample
[6]	4	0.972	cross-sample
[21]	4	0.980	not-spepcified
ConvNet	5	0.945	cross-finger
SVM	5	1.0	cross-finger
SVM	4	1.0	cross-finger

7. Conclusion

In this paper, we propose a deep learning approach for automatic fingerprint type classification. We design a deep ConvNet based on residual network. To preserve as much fingerprint details as possible, the input image is designed to be 512×512 and we can use two early convolutional layers to reduce the computational cost. The deep ConvNet serves as a feature extractor and on top of that a SVM is trained as the final classifier. Experiment results show that the proposed automatic that does not rely on any hand-crafted features can achieve high accuracy comparable to the state-of-the-art approach even if only label is used. Our proposed can accurately predict the fingerprint class using raw images, avoiding the need of using orientation filed estimation. Future works include using more advanced deep

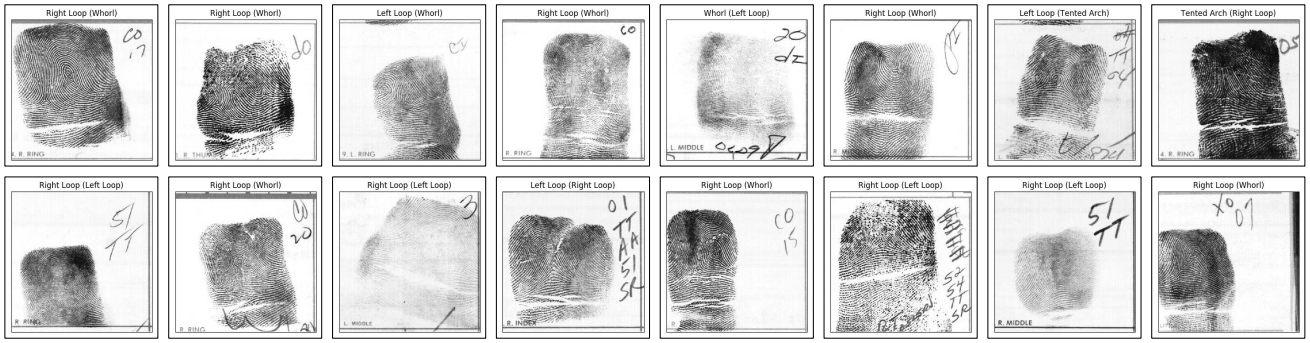


Figure 7: Mis-classification examples on NIST SD14. The title of each example is *Prediction*(*Ground Truth*)

networks and ensemble techniques to fuse multiple classifiers.

References

[1] Nist finger image quality 2.0. https://www.nist.gov/services-resources/software/development-nfiq-20.

[2] Unique identification authority of india. https://portal.uidai.gov.in/uidwebportal/dashboard.do.

[3] Nist special database 14, 2010. https://www.nist.gov/srd/nist-special-database-14.

[4] Nist special database 4, 2010. https://www.nist.gov/srd/nist-special-database-4.

[5] S. Bernard, N. Boujemaa, D. Vitale, and C. Bricot. Fingerprint classification using kohonen topologic map. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 3, pages 230–233 vol.3, 2001.

[6] K. Cao, L. Pang, J. Liang, and J. Tian. Fingerprint classification by a hierarchical classifier. *Pattern Recognition*, 46(12):3186–3197, 2013.

[7] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint classification by directional image partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):402–421, May 1999.

[8] R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[9] A. Fitz and R. Green". Fingerprint classification using a hexagonal fast fourier transform.

[10] F. Galton. *Finger Prints*. Macmillan, London, 1892.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] E. R. Henry. *Classification and uses of finger prints*. HM Stationery Office, 1905.

[13] A. K. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):348–359, Apr 1999.

[14] K. Karu and A. K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[18] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[20] B. T. Ulery, R. A. Hicklin, J. Buscaglia, and M. A. Roberts. Accuracy and reliability of forensic latent fingerprint decisions. *Proceedings of the National Academy of Sciences of the United States of America*, 108(19):7733–7738436, 2011.

[21] R. Wang, C. Han, Y. Wu, and T. Guo. Fingerprint classification based on depth neural network. *arXiv preprint arXiv:1409.5188*, 2014.

[22] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):918–930, 2016.