Deep Convolutional neural network for Fingerprint Pattern Classification

Anonymous IJCB 2017 submission

Abstract

Fingerprints are broadly categorized into five pattern class types. Fingerprint pattern classification is useful for quick exclusions and therefore reducing the search space by allowing for search and comparison only within same pattern class type. Fingerprint pattern classification can be done by visual examination of shape and characteristics of regions of a fingerprint; or using classification algorithms such as neural networks or K-nearest neighbors that are trained to recognize some specified patterns. To our knowledge, all of the current fingerprint pattern classification algorithms require some processing of fingerprint images to extract features, for example, estimate orientation flow and detect singularities. In this paper, we present deep learning approach for automated fingerprint pattern type classification without explicit characterization of fingerprint or any feature extraction. This method uses deep convolutional neural network(deep ConvNet) as feature extractor and a (Support Vector Machine) SVM as classifier. Our results show that the proposed approach outperforms the state-of-the-art approaches, achieving 98.61% *accuracy on NIST SD14 and* 100% *accuracy on NIST SD4.*

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, a system needs to search and compare the query print with the fingerprints stored in its reference (or enrolled) database. The size of a reference database can be from thousands to hundreds of millions of 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 Figure1. The Fingerprint Source Book [19] defines each of these patterns as follows: Arch is a pattern type in which the friction ridges enter on one side of the impression and flow, or tend to flow, out the other side with a rise or wave in the center. Tented arch is a pattern type that possesses either an angle, an upthrust, or two of the three basic characteristics of the loop. Loop is a pattern type in which one or more friction ridges enter upon one side, recurve, touch or pass an imaginary line between delta and core and flow out, or tend to flow out, on the same side the friction ridges entered. Loops can be left slant loops (or left loops), in which the pattern flows to the left in the impression; or right slant loops (right loops), in which the pattern flows to the right in the impression. Whorl is a fingerprint pattern type that consists of one or more friction ridges that make, or tends to make, a complete circuit, with two deltas, between which, when an imaginary line is drawn, at least one recurving friction ridge within the inner pattern area is cut or touched. Fingerprints collected in real world do not rigidly follow these definitions. In fact, many fingerprints can be classified into multiple types at the same time due to ambiguity.

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.[1] 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. Challenges in fingerprint pattern classification include: 1) quality of fingerprints, characterizing poor quality images are particularly difficult, 2) small

[1] Some of more recent fingerprint classification algorithms use continuous classifications instead of discrete classes. These continuous spectrums are beyond the scope of this paper because of their non-intuitive and proprietary nature as they are developed for a dedicated recognition algorithm.

inter-class dissimilarity and small intra-class similarity, for example, tented arch and loop may look similar; 3) ambiguities in some pattern class labels, some fingerprints can be classified into multiple classes, or different classes by different fingerprint experts.

In this paper we propose an automated fingerprint pattern classification that unlike previous finger pattern classifiers, does not require feature extraction. Specifically, we trained a deep ConvNet to take 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.

The rest of the paper is organized as follows. Section 2 overviews previous work. Section 3 details our technical approach. We describe our experimental set up, data and results in Section 4. Finally, we conclude in Section 5 and present our suggested way forward.

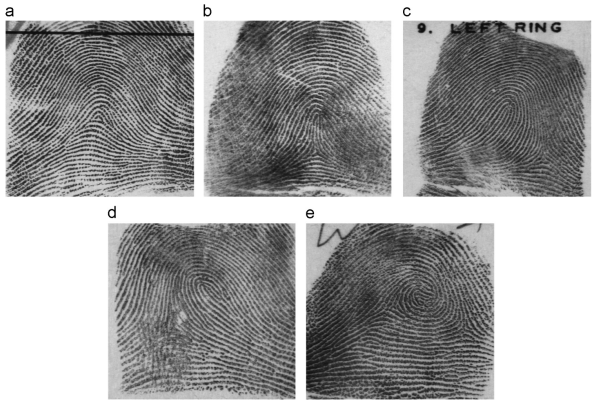


Figure 1: Examples of fingerprint classes[6]: (a) Arch, (b) Tented Arch, (c) Left Loop, and (d) Right Loop. Because arch and tented arch only accounts for a small portion in human, most often they are combined into one class.

2. Previous work

((I switched the first and scond paragrph)

Many existing works are focused on fingerprint type classification. 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.* [22] also used orientation filed as features.

Previous works mostly consist 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 pattern classification 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. Their repeatability and reproducibility cannot be guaranteed either, due to inter- and intra-examiners variations [21]. 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 exhibits powerful classification capability in many areas[17][20]. Different from common CNN-based approach, we employs a SVM on top of deep ConvNet as classifier for prediction.

3. Methodology

Figure2 illustrates our technical approach. The top row shows the steps of the training process. First training images are preprocessed using data augmentation techniques to increase data diversity. The augmented data are fed into deep ConvNet for training. The trained deep ConvNet then serves as a feature extractor for a SVM which uses the deep features from the trained deep ConvNet as input, and is trained to classify the pattern type. During the testing, shown in the bottom row of Figure2, testing images are fed into the trained deep ConvNet for deep features extraction. These intermediate deep features are used as input features to the trained SVM. The output of the SVM is final pattern class prediction.

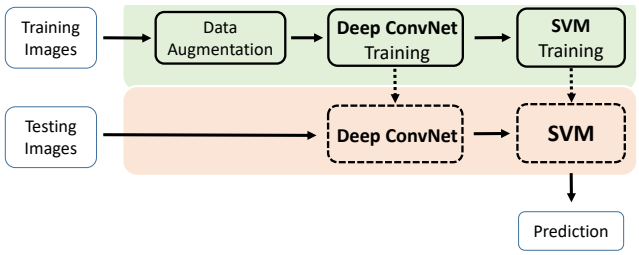


Figure 2: Overview of proposed approach.

3.1. Deep ConvNet Architecture

Our proposed network architecture (Figure 4) 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 summarizes 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 and explanations regarding network architecture can be found 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. This preserves the detail of fingerprint to the extent possible. However, the larger the size of input images, the larger the computational cost. We empirically searched for the optimal input image size and we found that the performance drops as the size gets smaller, and it drops significantly for images smaller than 224×224 – as shown in Figure3, down-sampling the images to 224×224, results in loss of information content of prints and reduces the clarity of ridge details and deltas and cores. Therefore we decided against significant down sampling. We also ruled out cropping or segmenting the fingerprint portion of an image, that is removing background and non-fingerprint portion of images to get smaller images, because of its requiring to employ a segmentation algorithm and reliance on the accuracy of the segmentation algorithm – recall that one of our objective has been to avoid fingerprint processing and characterization. After careful visual inspection of fingerprint images of different sizes, we decided to use 512×512 as input size for the following reasons. First, sufficient fingerprint detail is preserved at this size. Second, it is a reasonable size for rolled fingerprints. Finally, it is the original image size of one of the datasets we used, which is NIST SD4. To remedy the huge computational cost and high memory usage during the training, we added *conv1* and *conv2* with stride 2 to down-sample the input images. As shown in Table.1, after *conv2*, the feature map size is $128 \times 128, 96$. So, the spatial size is reduced (from 512×512 to 128×128) and spatial information is stored in the increased channels (from 1 to 96).

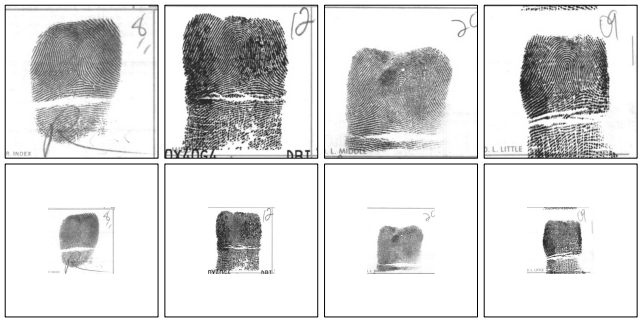


Figure 3: Example images from NIST SD14 with different spatial sizes. The top row images are 512×512 and the bottom row are 224×224. Ridges of top images are more distinguishable than that of bottom images.

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	$1 \times 1, 5$
	$1 \times 1, 5$	

3.2. Data Augmentation

Fingerprint images exhibit a wide range of location, rotation, brightness and contrast. To enhance the generalization ability of our ConvNet, we adopt data augmentation techniques to increase the data diversity.

To augment training dataset, we applied below augmentation techniques:

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

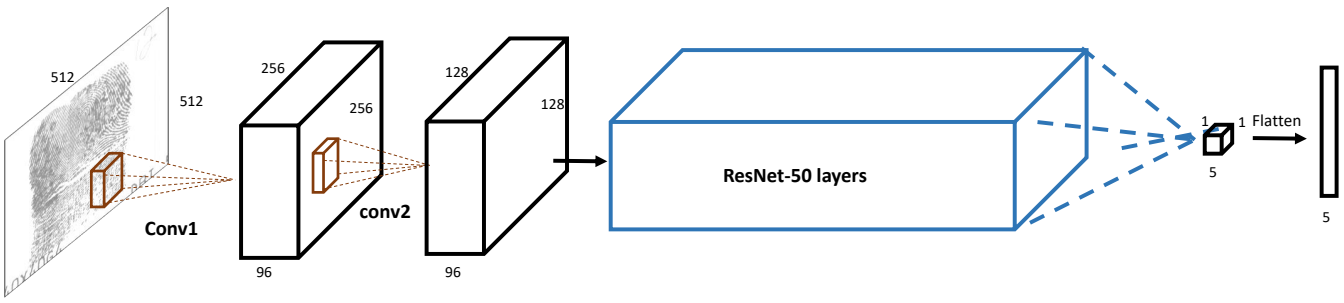


Figure 4: Architecture of proposed CNN.

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 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 $uniform(0.4, 1.6)$.

3.3. SVM

The deep ConvNet only serves as a feature extractor and we use a non-linear SVM as 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. Therefore, 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 pattern class types.

4. Experiments

4.1. Dataset

We used NIST Special Database 4 [4] and NIST Special Database 14 [3] for our experiments.

NIST SD4 contains 2000 8-bit gray scale fingerprint image pairs, a total of 4000 images. Each fingerprint pair consists of two different rollings of the same finger. The size of each image is 512×512 pixels and each image is classified using one of the five following classes: arch, left or right loops, tented arch, or whorl. The database is evenly distributed over each of the five mentioned classifications with 400 fingerprint pairs (800 images) from each class.

NIST SD14 contains 27000 8-bit gray scale fingerprint image pairs, a total of 54000 images. There are 2700 subjects in this dataset and each subject has 10 fingerprint pairs. The size of each image is 768×832. To fit in our network (see Section 3.1), we centrally crop each image into a 768×768 pixels image and then resize it into 512×512 pixels. Each image is classified using the five finger pattern types mentioned above. Table 2 shows the distribution of fingerprint pattern classes. The distribution of classes are supposed to approximate the natural horizontal distribution of the five pattern type classes. We can see that unlike NIST SD4, arch and tented arch samples are only a small portion of NIST SD14.

Table 2: Fingerprint Pattern Class Distribution of NIST SD14.

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

4.2. Experimental Setup

We used a i7-5930K desktop with 32GB memory and a Nvidia GTX TITAN X GPU for experiments. We used 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 used ℓ_2 regularization with 0.0001 weight decay rate. The batch size is 32. We ran two experiments using NIST SD4 and NIST SD14, respectively. Each experiment is trained for $20k$ steps.

For the NIST SD14 experiment, we used images of the 80% of the subjects for training, a total of 2160 subjects, 43200 images. Among these 432000 images, 36 of them have labels tother than the 5 classes mentioned above. These 36 images were discarded. Images from the remaining 20% of the subjects are used for testing, a total of 10800 images. 9 of these images were discarded due to the same reason above.

For the NIST SD4 experiment, we adopted two evaluation protocols. The first protocol is cross-sample, where we used the first samples of each fingerprint pair for training and the second samples for testing. This is the same

protocol used by previous works reported in the literature, and therefore allows for fair comparison of our results with related works. The second protocol is cross-finger, where we used images of 50% of the fingers for training and the other 50% for testing. This ensures images of the same finger does not exist in the training and testing sets at the same time. To improve the performance for the NIST SD4, we use NIST SD14 data to pre-train the model.

The following sections, report classification accuracy for 5-class fingerprint pattern type classification for two cases: a) SVM is trained as a classifier using deep features generated by our trained ConvNet, and b) ConvNet is used as the classifier. Additionally, we evaluated the performance of the SVM on 4-class fingerprint classification where arch and tented arch classes are combined. 4-class classification is also reported in other studies.

4.3. NIST SD14 results

Table3 summarizes our results for NIST SD14 images. ConvNet and SVM achieve similar accuracy (0.9861) for 5-class pattern type classification. However, ConvNet seems to perform slightly better in terms of average precision, recall rate and F1 score. Performance of the SVM classifier is slightly improved when arch and tented arch are combined into one class (4-class classification), achieving accuracy of 0.9875. The confusion matrices of the 5-class classification are shown in Figures 5a and 5b. Figure 6a shows similar results for the 4-class classification. Despite the unbalanced distribution of fingerprint pattern class types, as indicated by the small number of images of arch and tented arch pattern type compared to other pattern classes, our approach achieves high accuracy. The lowest precision (0.959) and recall (0.950) rates are observed for tented arch. We believe this is due to lack of sufficient training samples as well as possible label ambiguity. Figure 6a shows improved precision and recall rates (0.98), suggesting combing arch and tented arch classes can eliminate some cases of mis-classifications.

Table 3: Experiment results for NIST SD14. Columns 4, 5 and 6 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

Figure7 shows some examples of mis-classification cases for the NIST SD14.

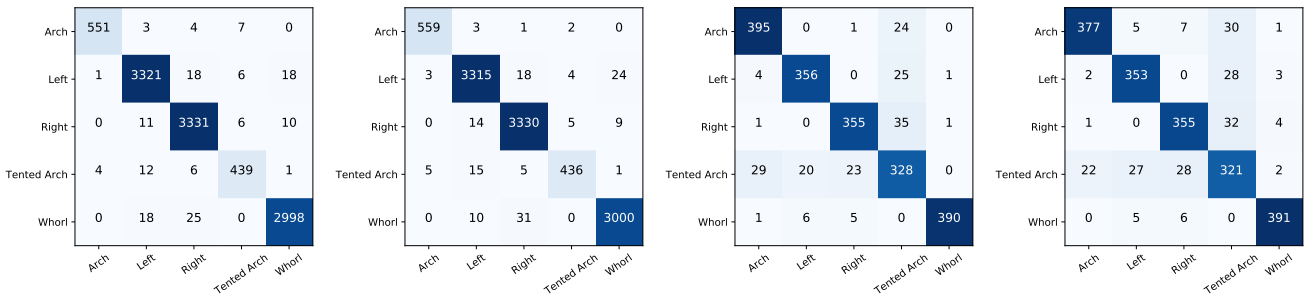
4.4. NIST SD4 Result

Table4 summarizes our results for NIST SD4 images. We make three observations from Table4. First, for both protocol, the 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. Our second observation is that cross-finger protocol shows lower performance compared to cross-sample. For 5-class ConvNet, the accuracy drops 0.023. SVM shows smaller performance drops: for 5-class SVM, the accuracy is 0.015 lower, and for 4-class SVM, the accuracy drops 0.011. The performance drop is small though, indicating the generalization ability of our proposed method. Thirdly, as we observed and discussed with NIST SD 14 results, the accuracy is improved when tented arch and arch classes 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 some mis-classification errors can be eliminated by combing arch and tented arch classes.

Figure5c and Figure5d show confusion matrices for 5-class classification using cross-finger protocol. Figure 6b shows similar results for 4-class classification and cross-finger protocol. Similar to NIST SD 14 results, tented arch gives the lowest precision (91.8%) and lowest precision rate (94.0%) among five classes. About 17% of the NIST SD 4 images have two class labels. That is because of the ambiguity in class types, the images were assigned primary and secondary pattern class types. Previous work used performance criteria such that classification into either primary or secondary class is counted as a correct classification. We followed similar performance criteria. The training procedure remains the same as before where we use only the primary pattern class labels for training. In testing stage, for the 17% of the samples with more than one class label, as long as the prediction matches one of the labels the test sample is considered as a correct classification. Results tabulated in Table5 shows a significant performance gain. Our proposed ConvNet achieves 0.9535 accuracy for cross-sample protocol, which is 0.032 higher than before. And 0.945 accuracy, for cross-finger protocol, which is 0.046 higher than before. The proposed 5-class and 4-class SVMs achieve 100% accuracy for both cross-finger and cross-sample protocols.

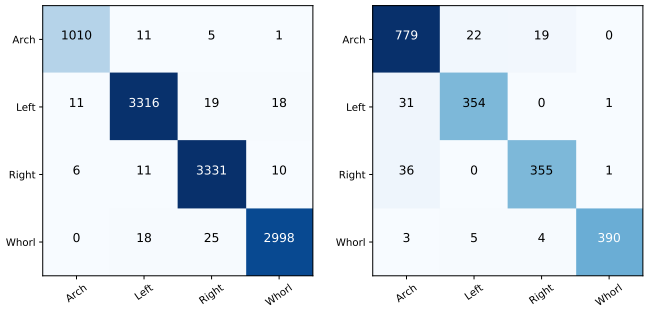
4.5. Discussion

Experiment results indicate that our proposed method can successfully perform fingerprint pattern class type classification on raw fingerprint images without any feature extraction, and, to the best of our knowledge, achieves better results than previously reported work. Using Deep Con-



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

Figure 5: Confusion Matrices for 5-class classification



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

Figure 6: Confusion Matrices for 4-class classification

Table 4: Experiment results for NIST SD4. Column 4, 5 and 6 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

vNet 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 pattern class types contributes the most error rate among the five classes. Combining arch and tented arch classes into one improves performance. Allowing more than one class assignment for the ambiguous cases result in further improvements.

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
[22]	4	0.980	not-specified
ConvNet	5	0.945	cross-finger
SVM	5	1.0	cross-finger
SVM	4	1.0	cross-finger

5. Conclusion and Future Work

We presented a deep learning approach for automated fingerprint pattern type classification. We designed a deep ConvNet based on residual network. To preserve as much fingerprint details as possible, the input image is designed to be 512×512 pixels and we used two early convolutional layers to reduce the computational cost. We did data augmentation to improve data diversity. The deep ConvNet serves as a feature extractor and a SVM is trained as the final classifier. Experiment results show that our proposed method achieves high accuracy comparable to the state-of-the-art approaches using raw images and without any feature extraction. We did not report reliability (or uncertainty) associated with accuracy of our method. We will include that as part of our future work by using different sampling of test and train sets. We would like to have tested our method on a larger datasets, but we were limited by the availability of ground-truth pattern class labels. We are seeking solutions for that. Also, as part of future work, we like to examine the robustness of our method particularly when dealing with images of poor quality. Other future works include using more advanced deep networks and ensemble techniques to fuse multiple classifiers.

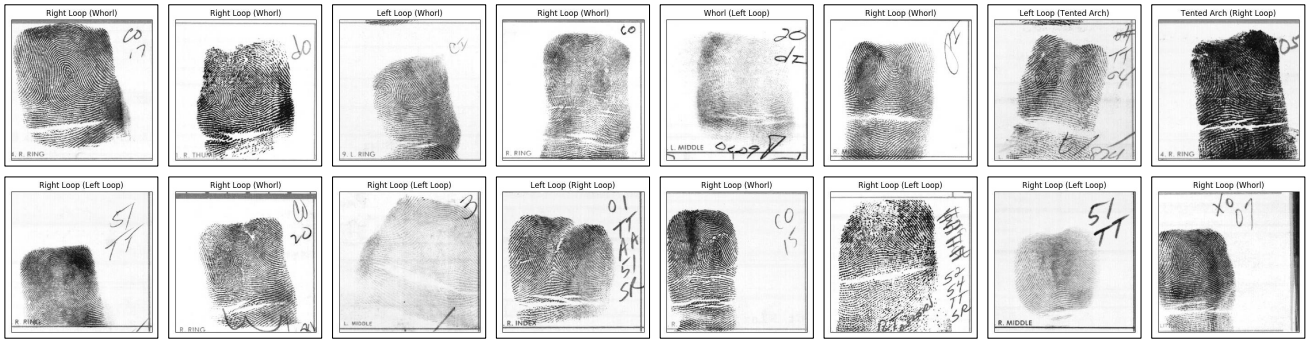


Figure 7: Examples of mis-classification cases for NIST SD14. Each example is labeled with *Prediction*(*Ground Truth*)

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] N. I. of Justice. *The Fingerprint Source Book*. US Department of Justice, 1892.

[20] 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.

[21] 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.

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

[23] 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.