# Deep Convolutional neural network for Fingerprint Pattern Classification

Elham Tabassi Xiao Zeng

## 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 [10]) and Sir Francis Galton [9] 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.

## 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.

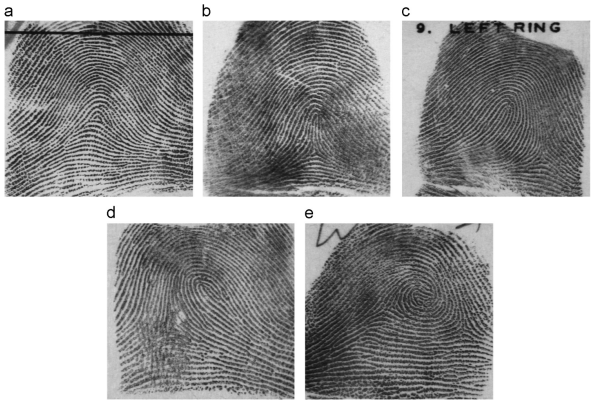


Figure 1. Examples of fingerprint classes[5]: (a) Arch (b) Tented Arch (c) Left Loop (d) Right Loop (e) Whorl.Note that tented and tented arch are similar.

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 [16]. 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[14][15].

An overview of related work follows. Karu and Jain [12] presented a rule-based classifier based on extracting singu-

lar points. Fitz and Green [8] used a Hexagonal Fourier Transform to classify fingerprints into whorls, loops and arches. Jain *et al.* [11] 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.* [6] 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.* [4] used the Kohonen topologic map for fingerprint pattern classification. Kai Cao *et al.*[5] proposed to extract fingerprint orientation feature and used a hierarchical classifier for classification. Ruxin Wang *et al.* [17] 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* [13]). 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)[7] 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[18]. 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 [3] 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.

## 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 4, 2010. https://www.nist.gov/srd/nist-special-database-4.

[4] 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.

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

[6] 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.

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

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

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

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

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

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

[18] 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.