COMP6706 Report: Towards More Accurate Fingerprint Recognition Using Deep Learning and Graph Matching

Name: Cao Pei
Student ID: 20031287r
Email: peipei.cao@connect.polyu.hk

Name: Feng Yulin
Student ID: 18070069r
Email: yulin.feng@connect.polyu.hk

Word Count

Abstract	A
Introduction	A
Related Work	A
Methodology	A
Experiments and Results	A
Conclusions and Future Work	A
Total (exclude references)	A

Abstract

This paper proposes a new deep learning based method for fingerprint recognition. A deep convolutional neural network is trained to extract local fingerprnt feature. This network can not only locate the local fingerprint region of interest, but also extract discriminative local feature, which is of significant importance for fingerprint matching. Graph neural network is used for fingerprint matching, which can robustly deal with fingerprint image noise. To our knowledge, this is the first work of applying graph neural network for fingerprint recognition. Conventional fingerprint matching uses hard coded matching algorithms, and those algorithms cannot compare with even minutiae pair based matching.

1 Introduction

The first time that fingerprint was used for human identification can be traced back to over one hundered years ago [1]. Since people found that no two individuals have the exactly same fingerprints, fingerprint starts to be used in forensics. For example, people who committed crimes might unintentionally leave their fingerprints in the crimes scenes which can later be identified as effective evidence and can help people find the criminals. Thus fingerprints starts to be used as one biometric for human identification and there has been so far a lot of research on human fingerprint recogntion. Compared with other biometrics such as face, iris and palmprint, fingerprint is highly distinctive, permanent, easy to accept and easy to collect [1], making itself one of the most popular biometrics which is today used in various recognition domains such as healthcare, E-business, border crossing and forensics.

Normally manual fingerprint recognition requires human experts to be invovled so that fingerprint can be matched against other fingerprint one by one. However, the cost of training such experts is high, and matching fingerprint with fingerprints in the database which main contain a huge number of registerd fingerprints is time consuming and slow, so the demand for an automated fingerprint recognition system is growing quickly. As a popular pattern recognition research problem, fingerprint recognition has attracted significant attention of both the academia and the industry. Although many fingerprint recognition algorithms have been proposed by different researchers over the past few decades and those algorithms showed their effectiveness in some fingerprint recognition domain, this is still a quite challenging problem especially for low quality fingerprints which happens often during acquisition due to unintended fingerprint noise and distortion. Therefore, fingerprint recogntion is an intriguing research problem even till today, and an accurate and fast fingerprint recognition algorithm is always preferred.

Generally speaking, the fingerprint images can be classified as 3 categories: plain fingerprint, latent print and rolled fingerprint [2]. The plain fingerprint images are usually acquired by directly touching the fingers on a flat surface. The latent images are usually collected from some real crime scenes, therefore these fingerprints are usually of low quality and contain a lot of noise. And the rolled fingerprint images are often collected by rolling the fingers from one side to another, which can keep most of the fingerprint ridge information [2].

There are seven kinds of main factors which is responsible for the intra-class variations: displacement, rotation, partial overlap, non-linear distortion, press and skin condition, noise and feature extraction errors [1]. The most important factor and also one of the most difficult problems is the non-linear distortion problem. The distortion is usually generated during the process of sensing the 3D fingerprint shapes onto a 2D flat surface. Most fingerprint matching algorithms usually do not directly deal with such issue, and treat the obtained fingerprint images as non-distorted by assuming that it was produced by a correct finger placement. Fingerprint recognition can be divided into two parts: fingerprint segmentation and fingerprint matching. Fingerprint segmentation is the process which extracts the foreground regions from the original fingerprint images [1]. There are many different kinds of segmentation methods, so we will use segmented images and will only concentrate on the matching part.

Fingerprint recognition algorithm, or fingerprint matching algorithm, can basically be divided into two categories: minutiae based matching and non-minutiae based matching. Fingerprint minutiae is considered as stable and robust local fingerprint ridge characteristics. Minutiae is usually represented by location, orientation, and minutiae type (i.e., ridge endings and ridge bifurcations). Therefore, minutiae based matching methods try to align two fingerprints by matching the largest number of corresponding minutiae pairs. For low quality fingerprint images whose minutiae are difficult to extract, non-minutiae based matching should be considered. Non-minuatia based matching usually includes some other fingerprint features such as ridge orientation, ridge frequency, and ridge shape. To date, most fingerprint recognition algorithms are based on minutiae because of its stability and robustness.

Conventional minutiae based fingerprint recognition method considers matching minutiae pairs by comparing minutiae between two fingerprints. The fast development and increasing popularity of deep learning methods shows remarkable progress in computer vision tasks. Some challenging biometric problems, such as face recognition [3] and iris recognition [4], have achieved great success thanks to deep learning based methods. There are also some works on using deep learning based methods for fingerprint matching and research shows that deep learning method can also improve fingerprint matching in some scenarios. As another popular research area, graph neural network has also made significant progress over the last decade. Since minutiae of one fingerprint can be formulated as a graph, this implies the possibility of applying graph neural network for matching minutiae graphs.

Our contribution in this work can be summarized as follows:

1. We proposed a new deep learning based model to accurately detect the fingerprint minutiae. We construct a well-designed minutiae heatmap and train a segmentation model to learn this minutiae map. The detection result of our model outperforms the previous deep learning based algorithms, such as MinutiaeNet [5] and non-deep learning algorithms, such as NBIS [6].

2. We propose a way to extract discriminative local fingerprint features, not restricted to fingerprint minutia, that can deal with fingerprint of different qualities. To achieve this target, we use deep learning based method, which shows its superiority in various computer vision tasks in recent years, to extract such feature. This deep neural can not only accurately locate the region of interest, but also learn discriminative local feature representation.

3. We propose graph neural network for fingerprint matching, which to our knowledge is the first work of applying graph neural network for fingerprint matching. The first thought is that fingerprint matching performance suffers from rotation, partial overlapping and non-linear distortion issues, and previous fingerprint matching algorithm cannot successfully deal with this issue. However, graph based matching is more robust to such noise. The second thought is although in the past there were some works of building fingerprint graph for matching, the performance is not as good as simple fingerprint pair or triplet matching.

4. Research on graph neural network has made great progress in recent years and has achieved success in many computer vision tasks. We hope our work on fingerprint recognition can make a contribution to the research on both fingerprint recognition and graph neural network.

There are some limitations of our work:

1. Since our work is based deep learning based method, which usually requires a lot of data to training the neural network, thus the success of our work relies on the amount of data and the quality of data we have. Deep neural work will show its power once more data is available. However, conventional hard coded matching conditions do not suffer from this problem.

2. Similar to other matching learning task, our work main also have over-fitting issue due to the limited amount of data that we have. Therefore, to mitigate this issue, a large number of experiments on different sources of fingerprint data should be made to validate the generality power of our matching algorithm.

This paper is organized as follows: we first discuss and compare related work on fingerprint recognition in Section 2. Section 3 presents our methods for fingerprint recognition and the corresponding experiment result is shown in Section 4. Conclusion and future work is given in Section 5.

2 Related Work

2.1 Conventional Fingerprint Recognition Method

Minutiae based method is the most widely used method for fingerprint recognition. Consider two fingerprint images with M and N minutiae respectively. Each minutiae is represented by location, orientation and type. A solution is required to match as many minutiae pairs as possible. To achieve this goal, local minutiae structures are formed to match minutiae. Compared with global minutiae matching [7], local minutiae matching are computationally cheaper and more robust against fingerprint distortion. Earlier approaches can be found in [8] and [9]. To describe local minutiae structure, two methods were proposed: nearest neighbor based and fixed radius based. For nearest neighbor based method, the number of neighbors is fixed [10] so each minutiae has same number of nearest neighbors. For fixed radius method, a radius is given so that all minutiae that fall within the circle defined by the minutiae as the center and the radius are included [11]. However, both approaches have their own drawbacks. Nearest neighbor method suffers from missing minutiae and spurious minutiae. Although fixed length method alleviates the issue of missing minutiae and spurious minutiae, it has potential border issues, which means that the minutiae near the border of the circle should be properly treated.

2.2 Deep Learning Method for Fingerprint Recognition

The past decade has witnessed the great progress of deep learning in computer vision and pattern recognition [12] [13] [14]. The name "deep learning" is derived from the architecture of deep artificial neural networks, which belongs to the family of machine learning methods. Compared with traditional neural networks which have only one or two layers, deep neural network can have tens or hundreds of layers. Such great progress of deep learning has also facilitated not only the research on fingerprint matching [15], but also the research on other fingerprint recognition related tasks, such as minutia extraction [16] [5]. [17] did the pioneering work on matching with contactless and contact-based 2D fingerprint with convolutional neural network (CNN), while [18] also used CNN to match 3D fingerprint, and both works showed that fingerprint recognition with CNN based method can achieve outperforming results. Instead of extracting minutiae for fingerprint matching, [19] trained a deep neural network to extract fixed-length fingerprint descriptor and regarded minutiae extraction as one step to achieve the final fixed-length representation.

2.3 Graph Neural Network

Minutiae of one fingerprint can be formulated as a graph by connecting each minutiae based on some predefined rules. However, due to fingerprint noise and distortion, minutiae graph matching can achieve as good performance as other minutiae based matching method [7]. There is another related research area called graph neural network (GNN).The success of deep neural networks has also boosted the research on graph neural network in recent years [20] [21] [22]. Compared with

images that are structured data, there are also unstructured data in the real world, such as social networks, molecules, and citation networks. Deep neural networks have some limitations to model such kind of data, but graph neural networks show their own flexibility. Motivated by CNN, there is a similar graph convolutional networks called GCN [23]. A graph is represented by nodes and edges. For classification task, it can be divided in to node level based classification and graph level based classification. GNN can also be used for graph matching [24] [25]. Graph matching is an NP-hard problem, and thus requires complex matching algorithm. [26] formulated graph matching as optimal transport problem which can be solved an efficient approximation algorithm called Sinkhorn algorithm [27].

3 Methodology (Research Project Details)

3.1 Minutiae Detection Network

We firstly detect the minutiae using deep learning models. Our basic idea is similar to [28], we first plot a circle with certain radius R (here we set as 10) for each minutiae and then set the pixels ini the circle as 1 and other pixels as 0. Then we improved it by setting the nearby pixels values as formula 1. Here r_i is the distance between the pixel's position to the *i-th* minutiae location, and R is a parameter which is used to adjust the circle size in the minutiae heatmap. It is easy to find that when r_i equals to 0, v_i will be 1, and when r_i equals to or great than R, v_i will be 0. Therefore, only the pixels within the distance of R of a minutiae has a positive value, and the value of the other pixels will be 0. If one pixel are in the nearby circles of two or more minutiae, the value will still be 1.

$$v = \min(v_i, 1) = \min(\sum_i \frac{1}{1 + e^{-0.5*(r_i - R)} - e^{R/2}}, 1) \tag{1}$$

Here we performed experiments using many kinds of deep learning structures: such as AutoEncoder, fully convolutional network (FCN) [29], DeepLab v3 [30] and UNet [31], and finally find that UNet can achieve the highest accuracy. Therefore we finally use UNet to locate the minutiae, and Fig. 1 presents the basic structure of our minutiae detection network. Similar to AutoEncoder, this network can be divided into two parts: encoder and decoder. The encoder downsample the augmented fingerprint images (with a size of $560 * 400$) to extract the feature while the decoder upsample those feature to generate a same size segmentation map. There are some shortcut connections between the encoder and the decoder, which can bypass more information to the decoder layer.

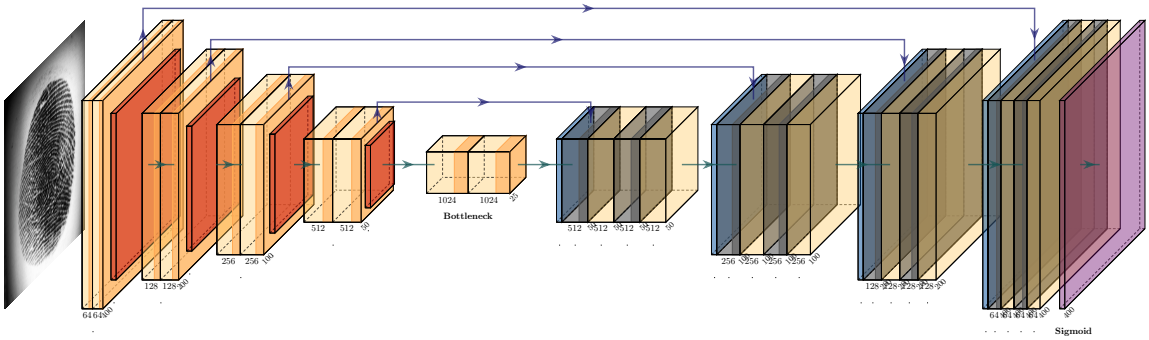


Figure 1: The structure of our minutiae detection network, plotted with [32]

We does not use normal UNet structure, but use the ResNet18 as encoder or backbone, which is similar to [33]. In addition, we use pre-trained weights in imagenet for the backbone initialization rather than train from scratch, because we find it takes much more time to converge when training from scratch. It should be noted that we output only one layer and use the sigmoid as the final layer's

activation function, which is mainly because that the output minutiae map is binary and it is not useful to use softmax function. Therefore, we use sigmoid function to limit the output probability into the interval of $[0,1]$, and then use binary cross entropy loss as the criterion loss function.

After calculate the minutiae map, we then need to locate the minutiae and output corresponding coordinates.

3.2 Minutiae Feature Network

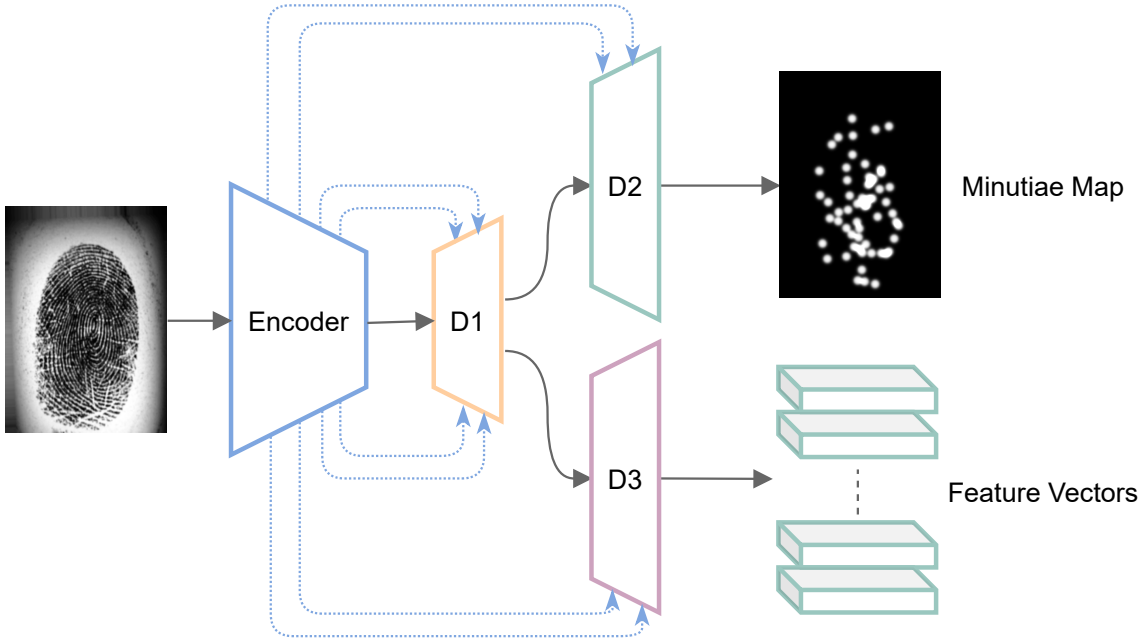


Figure 2: The structure of the minutiae feature network, Encoder and D1 is both used

After detecting the minutiae, we then need to calculate a feature vector for every minutiae. And we use unet to calculate the minutiae too. To save both the training time and computation time, we use a part of the minutiae detection network. Fig 2 presents the structure of our model. Encoder an D1 are shared between minutiae detection network and minutiae feature network. D2 are used to train the minutiae map while D3 are used to calculate the feature vector of every minutiae. Similar to minutiae detection network, the minutiae feature network also output a feature map (size is width*length*feature-channels) where every pixel has its own feature. Then we select a $32*32$ square patch centered on the minutiae point and calculate the average feature vector as the minutiae feature.

We trained the minutiae detection network first until it converges. Then we fixed the network weights of the Encoder and D1 and train D3 separately. Here we incorporated triplet loss function with the cosine distance during the network training. The triplet loss [3] was originally proposed in the face recognition area but it has shown to work well in other computer vision domain too.

This loss function can be defined as follows.

$$\sum_{i=0}^{n}[\mathscr{F}(x_i^a,x_i^p)-\mathscr{F}(x_i^a,x_i^n)+margin]_+ \tag{2}$$

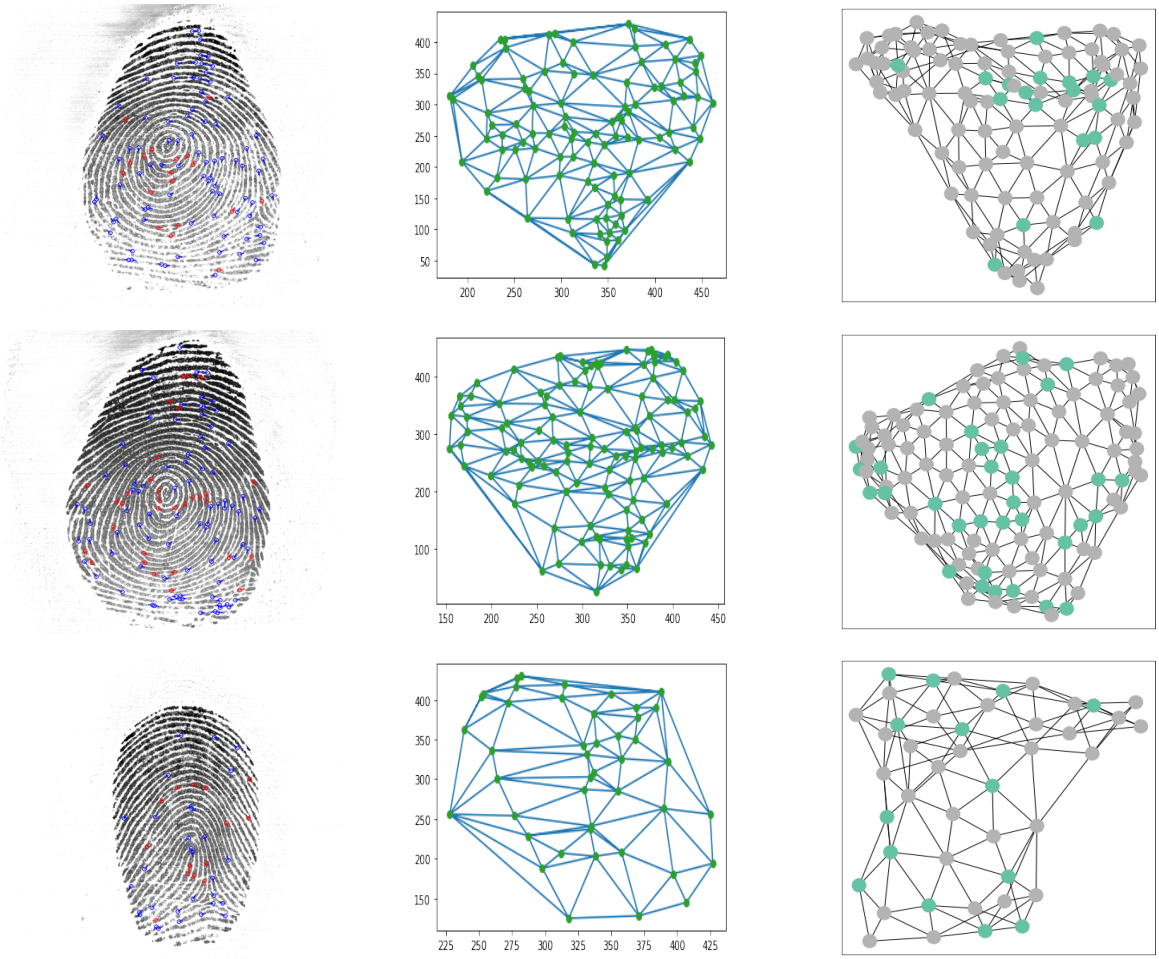
where x_i^a, x_i^p and x_i^n respectively represent the feature vector of anchor, positive and negative sub-images. While $\mathscr{F}$ is the cosine distance function between the two feature vectors as defined in the following.

$$\mathscr{F}(x^u, x^v) = \frac{\sum_{i=0}^{n}(x_i^u \times x_i^v)}{\sqrt{\sum_{i=0}^{n}((x_i^u)^2)} \times \sqrt{\sum_{i=0}^{n}((x_i^u)^2)}} \tag{3}$$

To construct triplet pairs for training, we first randomly select a minutiae from the dataset as anchor data, then we find its corresponding minutiae in another images of the same subject using Minutia Cylinder Code (MCC) [11] and use it as the positive minutiae, finally we randomly select a minutiae from fingerprint images of other subjects as the negative.

3.3 Minutiae Graph

To build minutiae graph, we need to determine vertex and edge for the graph. The most intuitive way to determine the vertex is by choosing the minutiae as the vertex. Here, each vertex is represented by the minutiae pixel location (x, y) from the original fingerprint image. For the edge configuration, there are two ways to connect the vertices by edges. One way is to connect each vertex with its k nearest neighbor and the number k is preset by the user. However, this method is sensitive to missing or spurious minutiae. The other way is to connect the vertices through triangulation. We use Delaunay triangulation method to form the edges here.



(a) original fingerprint iamge and detected minutiae

(b) minutiae connected by triangulation

(c) minutiae graph visualization

Figure 3: Sample fingerprint and its corresponding minutiae graph

Figure 3 illustrates the way we build the minutiae graph. Each row shows one sample fingerprint image with detected minutiae and its corresponding minutiae graph. The left column shows the original fingerprint with detected minutiae that are plotted on it. The middle column shows the minutiae

that are connected by Delaunay triangulation method. Note that because the origin of the fingerprint image is on the top left and the origin of the plot is on the bottom left, the minutiae that are plotted in the middle column are turned upside down compared with those minutiae in the left column. The right column shows the minutiae graph once the vertices and edges are built from the previous two columns. The geometric information of the minutiae is lost in the minutiae graph.

3.4 Data Augmentation

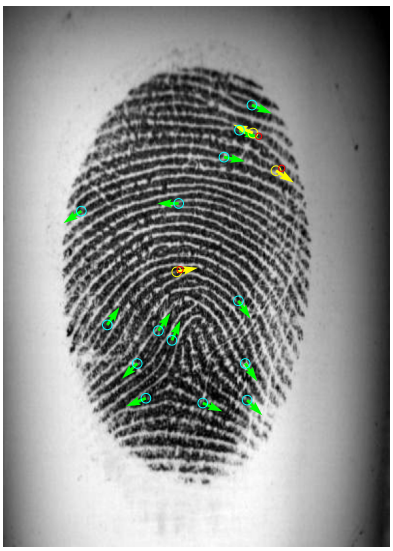
Figure 4: Augmentation images samples, left are the original images while right are the augmented images

It is worth to mention that we also do some data augmentation in our experiment to improve the robustness and accuracy. We use imgaug [34] library in this experiment. And mainly use the following transforms: random brightness, gamma contrast, Gaussian blur and shift scale rotate. After the data augmentation, we add a histogram equalization to normalize the images. Fig 4 presents some sample data augmentation images pairs, where the left is the original fingerprint image and the right is the augmented image.

4 Experiments and Results

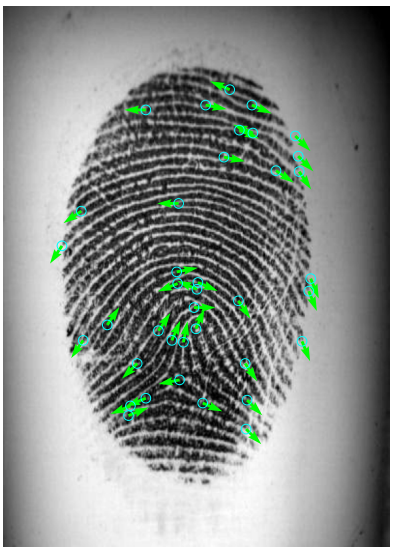
4.1 Training Data Preparation

We first tried to use NBIS [6] to mark the ground truth minutiae and use it for training. However, we found that the minutiae detection accuracy of NBIS is low. It will output an image quality number together with the minutiae location and we can set a threshold to select minutiae. However, the accuracy is low and if we set a high threshold, then many correct minutiae could not be detected, such as the image shown in Fig. 5a. And if we set a higher threshold, then it will detect many wrong minutiae, as shown in Fig. 5c. Besides, it cannot find an accurate minutiae threshold, as it either detected a lot of wrong minutiae or miss many important minutiae, or both. In addition, the locations of much minutiae were wrongly labeled. For example, in Fig. 5c, three minutiae's locations was not correct and we highlight them using yellow color, and we also mark the correct position with red circle. We can find that there are about 10 pixels gap between the correct minutiae location and those wrongly detected minutiae location, which will make it harder to train a good model.



(a) Minutiae detected by NBIS (with a high quality threshold)

(b) Our manually marked minutiae



(c) Minutiae detected by NBIS (with a low quality threshold)

Figure 5: A sample manually marked minutiae with corresponding NBIS detected minutiae

Therefore, we decide to mark the ground truth minutiae ourselves using labelme [35]. Because it is difficult to find some latent minutiae in the original image, we first enhance the original fingerprint image by estimating the orientation first and then binarizing with the orientations [36]. After that, we merge the binarized enhanced images with the original images, and then use labelme to manually mark the minutiae on the merged images. We did not directly use the binarization enhanced images because that there are much wrong minutiae in the enhanced images, therefore we should refer the original images too. Fig. 5b presents an example of the the merged images and our manually marked minutiae. It is the same image as Fig. 5c and 5a, where our manually marked minutiae is much more accurate than them.

As a result, we first manually selected 200 fingerprint images from the FVC2006 dataset [37] and marked them manually. Because we thought these images were not enough and therefore we also manually selected 300 more different high quality fingerprint images which minutiae detection are relatively accurate. These images are all very clear and the most minutiae was detected by minutiae. We set a high threshold (35) to filter the incorrect minutiae and get the ground truth minutiae (although there was some missed minutiae and little wrongly detected minutiae).

4.2 Minutiae Detection

We test our model using the rest of the fingerprint images in FVC2006 dataset. Fig 6 presents some randomly selected sample images and corresponding minutiae map. We also merge the original images and the predicted minutiae map to make it more clear to view. From Fig. 6, we can find that the minutiae in sub-figure 6f and 6i are very accurate. Almost every minutiae is detected and there is not wrongly detected minutiae. The minutiae in 6b and 6l is a little worse, where a few minutiae in the noisy area is wrongly detected.

The minutiae detection results are much better than existing algorithms, such as MinutiaeNet [5]. We used the source code provided in [5] and calculate the corresponding minutiae. Fig. 7 presents some sample minutiae detection results. We can easily find that there is a lot of wrongly detected minutiae and missed minutiae. Due to the time limitation, we did not implement other deep learning based algorithms and make comparison, but based on the detection results, we think our minutiae detection network achieve a very high accuracy and may surpass many existing minutiae detection algorithms.

(a) original image (b) minutiae map (c) merged image (d) original image (e) minutiae map (f) merged image

(g) original image (h) minutiae map (i) merged image (j) original image (k) minutiae map (l) merged image

Figure 6: Some sample images and predicted minutiae map

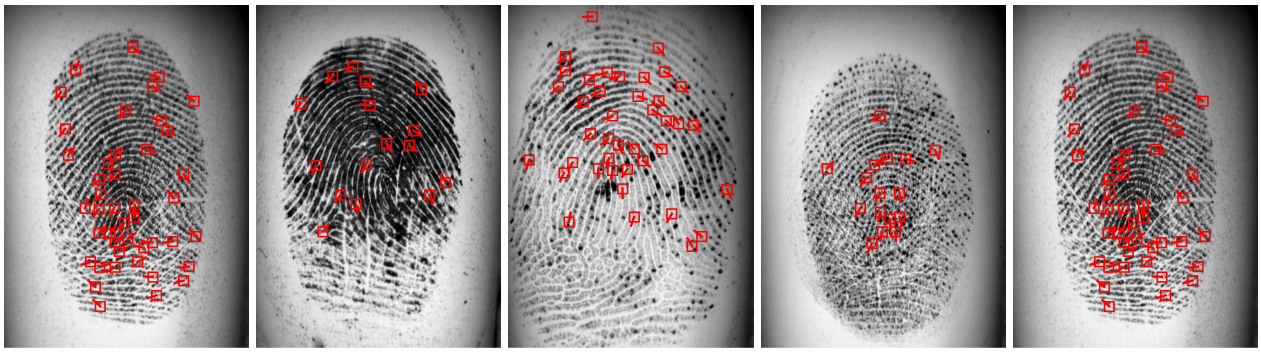


Figure 7: Some sample minutiae detection results of MinutiaeNet [5]

4.3 Graph Triplet

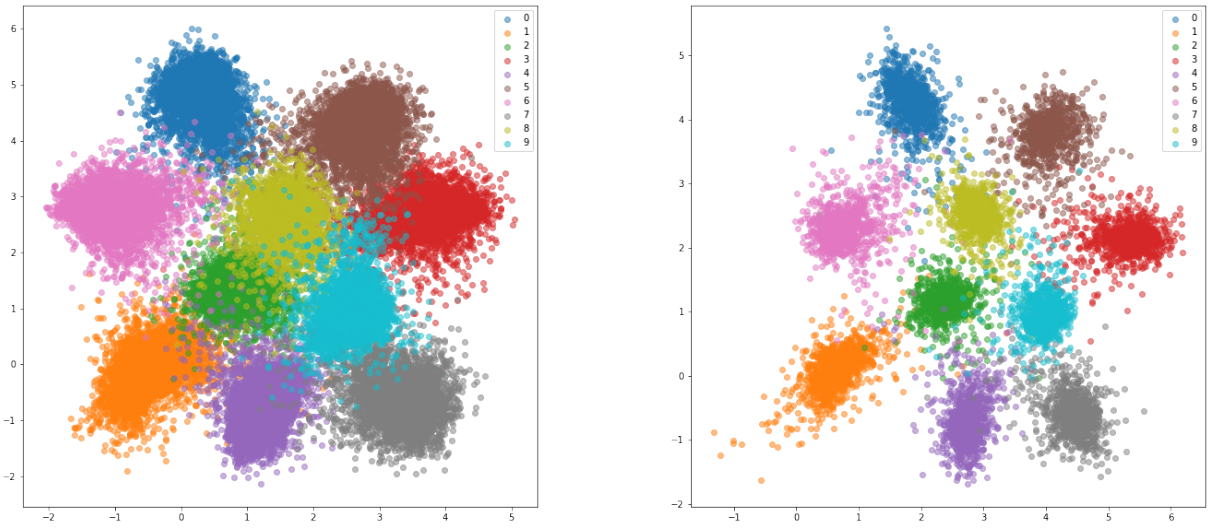


Figure 8: Visualization of MNIST training set and test set result

We use MNIST dataset [38] to build the graph triplet, and use SplineCNN layer [39] [40] to replace the CNN layer which is normally used in computer vision task. Each MNIST image is represented as a graph, whose nodes correponds to the pixels in the image and the edge is formed by connected each pixel to all its neighboring pixels. Therefore, based on this setup, each MNIST image is a graph with the same number of nodes, which is $784 (= 28^2)$. The graph triplet is formulated as followes: for each sample as the anchor in the training dataset, we randomly select another sample which belongs to the same class as the positive sample, and randomly select one sample from the rest classes as the negative sample.

Similar to CNN neural network configuration, we build a SplineCNN network, which contains two SplineCNN layers, and ech SplineCNN layer is followed by an exponentail linear unit (ELU) and a maxpooling layer. There are two fully connected layers, the first fully connected layer is also followed by an ELU layer, the output dimension of the final fully connected layer is 10 (we also choose the final dimension to be 2 for visualization). The initial learning rate is 10^{-3}. Stochastic gradient descent with Adam method is used for optimization and the total training epochs is 20.

For evaluation, since the total number of samples of MNIST test dataset is $10,000$ and they are not balanced for each class, we randomly choose 500 samples from each class. We use all-to-all evaluation protocal, which means for each test sample, Euclidean distance is computed against all the rest samples. Therefore, in total, we generate $1,247,500$ genuine scores and $11,250,000$ impostor scores. Since MNIST dataset is a simple dataset with a few number of classes and a large number of samples for each class, it turns out that there is no overlap between genuine scores and impostor scores. The equal error rate(EER) is thus zero.

5 Conclusion and Future Work

In this paper, we have

There are several directions that deserve to explore in the future:

References

[1] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer, London, 2nd ed edition, 2009.

[2] Rohan Nimkar and Agya Mishra. Fingerprint Segmentation Algorithms: A Literature Review. *International Journal of Computer Applications*, 95(5):20–24, June 2014.

[3] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[4] Zijing Zhao and Ajay Kumar. Towards more accurate iris recognition using deeply learned spatially corresponding features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3809–3818, 2017.

[5] Dinh-Luan Nguyen, Kai Cao, and Anil K Jain. Robust minutiae extractor: Integrating deep networks and fingerprint domain knowledge. In *2018 International Conference on Biometrics (ICB)*, pages 9–16. IEEE, 2018.

[6] Craig I Watson, Michael D Garris, Elham Tabassi, Charles L Wilson, R Michael McCabe, Stanley Janet, and Kenneth Ko. User's guide to NIST biometric image software (NBIS). Technical Report NIST IR 7392, Gaithersburg, MD, 2007.

[7] Sharat Chikkerur, Alexander N Cartwright, and Venu Govindaraju. K-plet and coupled bfs: a graph based fingerprint representation and matching algorithm. In *International Conference on Biometrics*, pages 309–315. Springer, 2006.

[8] Andrew K Hrechak and James A McHugh. Automated fingerprint recognition using structural matching. *Pattern Recognition*, 23(8):893–904, 1990.

[9] Zsolt Miklos Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

[10] Xudong Jiang and Wei-Yun Yau. Fingerprint minutiae matching based on the local and global structures. In *Proceedings 15th international conference on pattern recognition. ICPR-2000*, volume 2, pages 1038–1041. IEEE, 2000.

[11] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2128–2141, 2010.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] Kai Cao and Anil K Jain. Automated latent fingerprint recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):788–800, 2018.

[16] Yao Tang, Fei Gao, Jufu Feng, and Yuhang Liu. Fingernet: An unified deep network for fingerprint minutiae extraction. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 108–116. IEEE, 2017.

[17] Chenhao Lin and Ajay Kumar. A cnn-based framework for comparison of contactless to contact-based fingerprints. *IEEE Transactions on Information Forensics and Security*, 14(3):662–676, 2018.

[18] Chenhao Lin and Ajay Kumar. Contactless and partial 3d fingerprint recognition using multi-view deep representation. *Pattern Recognition*, 83:314–327, 2018.

[19] Joshua James Engelsma, Kai Cao, and Anil K Jain. Learning a fixed-length fingerprint representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[21] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[24] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2684–2693, 2018.

[25] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning*, pages 3835–3845. PMLR, 2019.

[26] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.

[27] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[28] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards Accurate Multi-Person Pose Estimation in the Wild. In *CVPR*, pages 4903–4911, 2017.

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, pages 3431–3440, 2015.

[30] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587 [cs]*, December 2017.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241. Springer International Publishing, 2015.

[32] Haris Iqbal. HarisIqbal88/PlotNeuralNet v1.0.0. Zenodo, December 2018.

[33] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[34] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[35] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[36] Letian Cao and Yazhou Wang. *Fingerprint Image Enhancement and Minutiae Extraction Algorithm*. 2017.

[37] Raffaele Cappelli, Matteo Ferrara, Annalisa Franco, and Davide Maltoni. Fingerprint verification competition 2006. *Biometric Technology Today*, 15(7-8):7–9, 2007.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[40] M. Fey, J. E. Lenssen, C. Morris, J. Masci, and N. M. Kriege. Deep graph matching consensus. In *International Conference on Learning Representations (ICLR)*, 2020.