Detecting Locally, Patching Globally: An End-to-end Framework for High Speed and Accurate Detection of Fingerprint Minutiae

Yulin Feng and Ajay Kumar

***Abstract*—Billions of fingerprint images are acquired and matched to protect the national borders and in a range of egovernance applications. Fast and accurate minutiae detection from fingerprint images is the key to advance fingerprint matching algorithms for large-scale applications. However, currently available fingerprint minutiae extraction methods are not accurate and fast enough to support such large-scale applications. This paper proposes a new method that uses a lightweight pixelwise local dilated neural network to extract local features and a patchwise global neural network to recover the global features. It consolidates the local and global fingerprint features to generate a full-size minutiae location map, and then accurately localizes the minutiae positions by using a recursive connected components algorithm. We design a new loss function to accurately detect minutia orientation and incorporate a dynamic end-to-end loss to provide effective supervision in learning discriminant features. It is due to the proposed design and loss function that can enable higher accuracy with significantly less computations. We present reproducible experimental results from five publicly available contact-based and contactless databases that indicate significant improvement in the minutiae detection accuracy, which also leads to enhanced fingerprint matching accuracy. Since the minutiae represent key points in the fingerprint images, the proposed end-to-end minutiae detection method also has a potential to be employed in many other key points detection tasks.**

I. INTRODUCTION

Fingerprint recognition offers significant accuracy and user convenience, making it highly attractive for a range of civilian applications. Over the past forty years, the automatic fingerprint identification has achieved success for a wide range of civilian, business and law-enforcement applications [2]. Accurate and fast extraction of fingerprint minutiae is a critical part of many fingerprint recognition algorithms [3]-[8]. However, automatic fingerprint minutiae extraction can be an extremely difficult problem for some low-quality fingerprint images. For example, Paulino et al. [9] propose a method which relies only on manually marked minutiae due to the poor performance of existing minutiae extraction algorithms. There are mainly two kinds of minutiae extraction algorithms: conventional algorithms and deep learning-based algorithms.

Conventional fingerprint minutiae extraction methods, e.g., [10]-[13], usually detect the minutiae by using handcrafted features that are designed based on the domain knowledge. These minutiae extraction methods usually consist of several procedures, including fingerprint segmentation, enhancement, thinning, and binarization. These conventional methods can achieve impressive performance on high-quality fingerprint images. However, they are not accurate enough for a range of large-scale or real-world applications that frequently present low-quality fingerprint images.

The authors are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csylfeng@comp.polyu.edu.hk; csajaykr@comp.polyu.edu.hk)

A. Related Work

During the past few years, deep neural networks have shown its potential in many fields, which have encouraged researchers to propose deep learning-based fingerprint minutiae extraction models to address the limitation of traditional methods. Authors in [14] uses an AutoEncoder network to distinguish a minutia and non-minutia patches from a large number of fingerprint patches. Reference [15] use deep scattering network [16] to extract minutiae feature. Researchers in [17] propose a new patch-based approach which uses a convolutional neural network to automatically learn to focus on minutiae points. Literature [18], [19], [20], and [21] use deep networks to extract fingerprint features with a sliding window/patch and then predict the probability for every window/patch. As this sliding window procedure needs to process a lot of windows, these methods are quite time-consuming. References [19], [22], incorporate domain knowledge into the deep neural network, achieving some success and improving accuracies on several public fingerprint datasets. However, these methods are even more time-consuming as they require complex processing and use neural networks with high computational complexity. The authors in [23] improve the network architecture and loss function, and propose a two-stage deep learning-based method. Besides, this method uses ROIAlign operation [24] to extract features for each patch. This method shows good performance in several contact-based fingerprint databases. However, there are still some problems with this algorithm. This method is not fast enough due to the two-stage network design and improper loss function for the orientation estimation. Reference [25] propose an UNet-architecture model to locate the minutiae and predict the orientation in contactless fingerprint images. Similar to [23], this method also extracts a coarse minutiae feature map and then determines the accurate minutiae by using another network. This method achieves good results in several contactless fingerprint databases. However, in addition to inadequate loss function design in the two-stage networks, it is slow due to the UNet-architecture and deep layers. The average minutiae extraction time by using a Titan Xp GPU card is 0.86 s, making it hard to be used in many real-world applications.

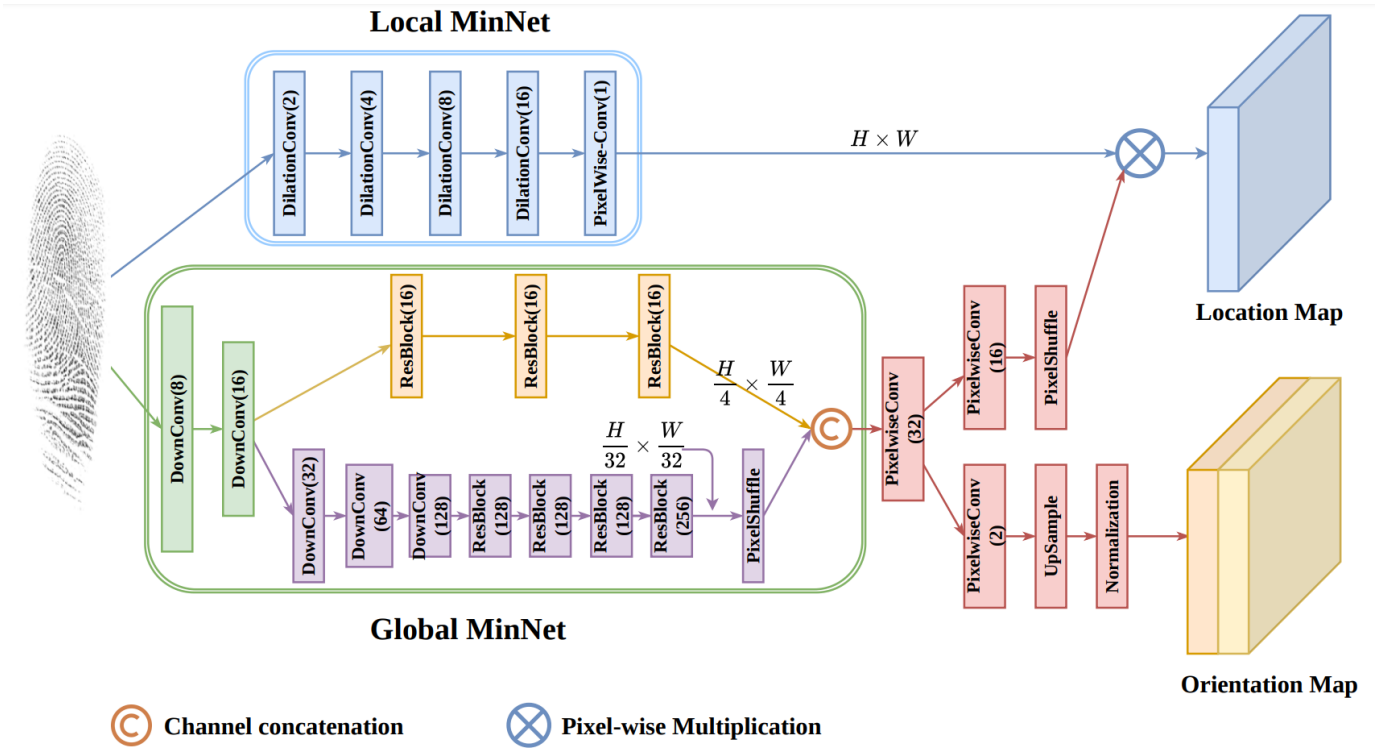


Fig. 1: Network architecture of proposed framework. "DilationConv" refers to dilated convolutional network."DownCon" is a convolutional layer with a stride of 2. "ResBlock" refers to the residual block. "PixelwiseConv" is a convolutional layer with a kernel size of 1×1."PixelShuffle" refers to the pixel shuffle layer. The numbers in the brackets represent the output channels of corresponding blocks.

B. Our Work and Contributions

This work introduces a novel neural network-based method for the fast and accurate fingerprint minutiae detection. We use a shallow dilated convolutional neural network to extract the pixel-wise local features and a deep neural network to extract patch-wise global features. We combine the global and local features to generate a pixel-wise minutiae location map, and then accurately locate minutiae positions by using a recursive connected components algorithm. It is well-known that the size of the training database used can greatly influence the deep learning model's effectiveness, and the model trained on a much bigger database generally performs better than the model trained using a smaller database. Several new synthetic fingerprint generation algorithm has been published during recent years, which makes it easy to generate a huge amount of fingerprint images. As a result, we introduce a new pipeline to train the deep learning model: we first generate a great amount of synthetic fingerprint images using [26], then pretrain deep learning model on such Generative Adversarial Network (GAN) generated fingerprint database, after that, we fine-tune the model on the training database. By adopting such generated fingerprint database for training, we greatly increase the detection accuracy. Besides, we improve the loss function and introduce a new end-to-end loss. Our approach can also automatically discard some targeted pixels during the training process, which speed up the training process and improve the performance. Due to such novel aspects in our design, the proposed method can achieve higher accuracy with much faster speed.

The contributions from this paper can be briefly summarized as follows:

1) A more accurate, robust, and faster neural network-based minutiae extraction algorithm is proposed. We present comparisons with state-of-the-art methods to show outperforming results for both the detection accuracy and the detection speed.
2) We propose a new deep neural network which consists of a lightweight pixel-wise local neural network and a deep patch-wise neural network. Compared with the UNet architecture (e.g., [25]), it has far less computational complexity, therefore it is much faster extracting the fingerprint features.
3) The loss functions for the fingerprint minutiae orientation estimation used in state-of-the-art algorithms are either not continuous or very slow to converge, we addressed these problems and achieved improved results by proposing a new loss function. We theoretically compare different kinds of loss functions used in state-of-the-art algorithms, to establish the merit of the proposed loss function. The final experiment results also support such conclusion or theoretical arguments.
4) We use a dynamic end-to-end loss in our experiments to jointly train the minutiae location and orientation networks. By dynamically changing the loss function, we firstly concentrate on training the minutiae location map and then focus on optimizing the loss function for

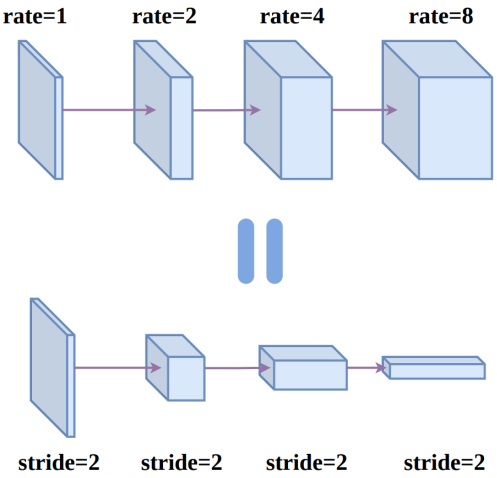


Fig. 2: The dilated convolutional network used in the proposed local MinNet and its corresponding ordinary convolutional network.

the orientation estimation. This kind of end-to-end loss function design may also be used in many other deep learning tasks.

5) Different from previous methods, the proposed approach can automatically discard some pixels during the training process rather than using the entire position heatmap, which helps to boost the training process and improve the accuracy. We also demonstrate the effectiveness of our pre-trained network with the GAN generated synthetic fingerprint images to achieve enhanced minutiae detection performance.

The rest of this paper is organized as follows: Sec. II details the methodology of the proposed approach, including the novel neural network design, proposed orientation estimation method, the ground truth minutiae map generation approach, the dynamic end-to-end loss function design, and the minutiae location estimation method; Sec. III introduces the experimental configurations, including the details of the databases used in this paper, and provides both the detailed experimental results and the analysis on these results; Sec. IV provides ablation study, sample visualizations and computational complexity analysis. Finally, the key conclusion from this work are summarized in Sec. V.

II. Methodology

A. Network Design

In order to extract a fingerprint minutia, we need to compute both its location and orientation. To compute minutiae locations, most previous deep learning-based methods output a minutiae map and then postprocessing the minutiae map to locate minutiae. In order to conserve the computation, the minutiae maps generated by these methods usually have smaller size than original fingerprint images. For example, [19] generates a minutiae map of size $\frac{H}{8} \times \frac{W}{8}$ (H and W are the height and width of the input fingerprint images respectively), [27] learns a minutiae map of size $128 \times 128 \times 6$ to encode an input fingerprint image of size $428 \times 428 \times 1$, while [23], [25] extract a minutiae map of size $\frac{H}{4} \times \frac{W}{4}$.

Approaches used in the literature [19], [23], [25], [27] can greatly reduce computations by reducing the size of the minutiae map. However, it has two main disadvantages. First, it cannot predict accurate minutiae positions when only using the minutiae map. Second, it may miss some minutiae that are closer to each other. In fingerprint images with a lot of minutiae, the distances between some minutiae are within 10 pixels. When reducing the minutiae map size to $\frac{1}{4} \times \frac{1}{4}$ of the original fingerprint image size, the distance between the new minutiae will be about 2 pixels, and it is quite possible that these minutiae will be regarded as one minutia. Therefore, these algorithms need some postprocessing to locate the minutiae. For instance, [23] uses a new network to predict the final location and orientation with some $7 \times 7 \times 256$ patch features. This postprocessing or second stage of operations can address such problem, while it still introduces a lot of additional computational complexity. Besides, it has two separate neural networks which are harder to optimize.

In order to address such intrinsic limitations, we propose a new end-to-end network architecture that can extract pixel wise minutiae maps within extremely low computational complexity. The main idea is to use a deep network (global MinNet) to extract global features and predict patch-wise minutiae maps and a shallow network (local MinNet) to extract local features and predict pixel-wise minutiae maps. The patch-wise minutiae map predicts accurate probabilities but is coarse-grained. In contrast, the pixel-wise minutiae map is fine-grained but is not sufficiently accurate due to shallow network and limited local features. Therefore, we upsample the patch-wise minutiae map by using pixel shuffle [28] and then multiply the pixel-wise location map to get a fine-grained and more accurate map. The complexity of a neural network mainly depends on two parts: network depth and channels number of every layer, and the input feature size of every layer. Since we only compute the patch-wise feature with the deep global MinNet and the pixel-wise feature with the lightweight local MinNet, it requires much less computations than those for computing pixel-wise feature with deep network.

Fig. 1 presents the entire network architecture. To extract the local minutiae map, we use a dilated convolutional neural network [29] as presented in Fig. 2. We firstly use a four-layer dilated convolutional network with the atrous rate of 1,2,4 and 8 respectively to extract the feature from every pixel, then use a point-wise convolutional layer to predict the probability from every pixel location. We use such a kind of atrous rate because these three layers equal to a conventional four-layers network with a stride of 2 (Fig. 2). The field is respectively $3, 7, 15$ and 31 after each dilated layer. Therefore, the local MinNet equals to a neural network that inputs a 31×31 patch and then predicts the probability of the center pixel. To extract the global minutiae feature, we first downsample the feature size by using two convolutional layers with the stride of 2. Suppose the original image size is $H \times W$, after these two downsample convolutional layers, it generates a reduced feature map with the size of $\frac{H}{4} \times \frac{W}{4}$. Next, we split the global MinNet into two branches. For the first branch, we use three residual blocks [30]

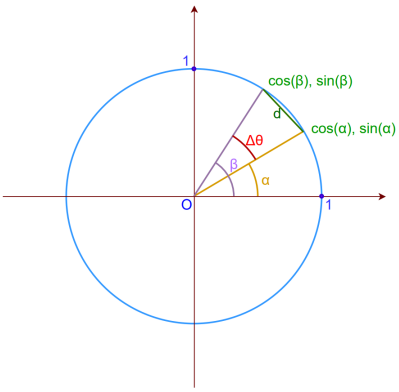


Fig. 3: The illustration of different loss function for the orientation estimation.

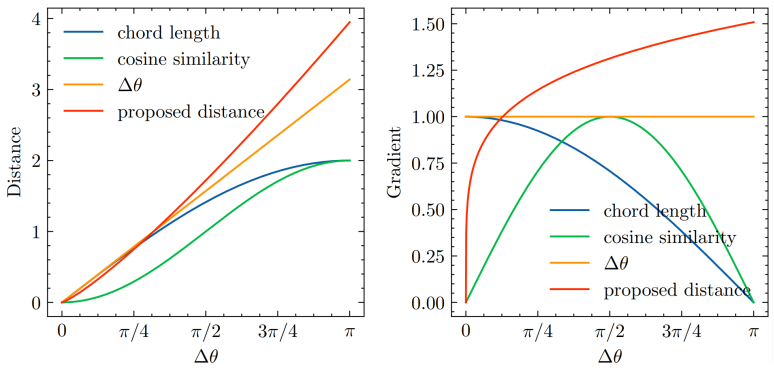


Fig. 4: Comparison of different loss function for the orientation estimation (left) and corresponding gradients (right).

to generate a relatively large feature map (feature map A) with the size of $\frac{H}{4} \times \frac{W}{4}$. For the second branch, we downsample the feature further and generate a global feature with the size of $\frac{H}{32} \times \frac{W}{32}$, and we add another four residual blocks to increase the network depth further and generate a more effective feature map (feature map B). After that, we upsample feature map B using a pixel shuffle layer [28] and it with feature map A. We use pixel shuffle layer, instead of direct upsampling, due to the following two reasons: first, it can yield a better performance; second, it decreases the feature channels and greatly reduces the computation complexity. We then upsample the feature to the original size and perform pixel-wise multiplication with the local minutiae map to generate the final location map.

After predicting the minutiae location map and detect all candidate minutiae, we predict the orientation map based on global features. We upsample it and then normalize the two orientation layers to cosine and sine value. We do not employ pixel-wise local features because it is relatively easy to predict the orientation and the global patch-wise features is already enough to predict accurate orientation. If we apply the pixelwise local features, we need to upsample the patch-wise global features first and then predict the orientation map, which will introduce a lot of additional computation.

By using such a network, we can predict pretty accurate pixel-wise minutiae map with a relatively low complexity.

B. Orientation Estimation

After locating the spatial location of the minutiae, it is also essential to accurately to compute their orientations. The orientations are not continuous, they are from 0 to 2π, while orientation 2π is the same as orientation 0. Therefore, it is not a good choice to directly predict the orientation by using the neural network. Reference [23] directly estimates the orientation. The authors normalize the orientation to $[-1, 1)$ and design a complex loss function to address this problem. Although it can solve the discontinuous problem, it is not straightforward and the loss function is complex. Therefore, instead of directly predicting the orientation, we predict the cosine and sine values of the orientation angles, and then convert them to the orientations.

Reference [25] also predicts cosine and sine value firstly. Suppose the ground truth orientation is α and the predicted orientation is β, and $\Delta\theta(\Delta\theta = |\alpha - \beta|)$ is the orientation difference between the ground truth and predicted orientation. As shown in Eq. (1) and Fig. 3, the loss L in reference [25] is computed using the Euclidean distance between points $(\cos\alpha, \sin\alpha)$ and $(\cos\beta, \sin\beta)$, which is equal to the chord length between orientation α and β in a unit circle.

$$\begin{aligned} L &= \sqrt{(\cos\alpha - \cos\beta)^2 + (\sin\alpha - \sin\beta)^2} \\ &= 2 \cdot \sin\frac{|\alpha - \beta|}{2} \\ &= 2 \cdot \sin\frac{\Delta\theta}{2} \end{aligned} \tag{1}$$

$$\frac{\partial L}{\partial \Delta\theta} = 2 \cdot \frac{\partial \sin\frac{\Delta\theta}{2}}{\partial \Delta\theta} = 2 \cdot \cos\frac{\Delta\theta}{2} \cdot \frac{1}{2} = \cos\frac{\Delta\theta}{2} \tag{2}$$

We compute the gradient of this loss function, which is shown in Eq. (2). We also plot the loss function and gradient in Fig. 4, from where we can observe that with the increase of $\Delta\theta$, the loss increases from 0 to 1, but the gradient decreases from 1 to 0 . As a result, when $\Delta\theta$ is large, the gradient is small, the network is trained slowly; and when $\Delta\theta$ is small, the gradient is large, the network continues the training and is not easy to converge. Specially, when $\Delta\theta = \pi$, the gradient reduces to 0 which means that the network stops backpropagation. Therefore, this loss function cannot be considered as a proper loss definition.

In order to address such limitations, we explore other distance functions and propose a new loss function. It is easy to observe that the root cause of this loss function's problems is that the backpropagation gradient decreases with the increase of orientation difference. This inspires us to find some loss functions which do not have such problems. First, we consider cosine similarity function, which is widely used in many deep learning models. However, it is not suitable for this task. As shown in Fig. 4, the gradient of cosine similarity function decreases from 1 to 0 in the region $[\pi/2, \pi]$. Next, we consider directly using the orientation difference $\Delta\theta$ as the loss function. Compared with chord length or cosine similarity, it is a better loss function. As shown in Fig. 4, the distance

increases along with the increase of $\Delta\theta$, while the gradient remains 1 , which can ensure faster training of the network. However, in this task, an invariable gradient is not desirable. Therefore, we address such limitations and propose a new distance function: $\Delta\theta^{1+\delta}$, where δ is a parameter, and is set as 0.2 during all our experiments. The gradient of this loss function is $(1+\delta)\cdot\Delta\theta^{\delta}$. As presented in Fig. 4, both the distance and gradient increases along with the increase of $\Delta\theta$. Besides, when the orientation difference decreases to 0, the gradient also decreases to 0, enabling the network to converge easily.

$$L_{\text{proposed}} = (\arccos(\cos\alpha\cdot\cos\beta+\sin\alpha\cdot\sin\beta))^{1+\delta}. \quad (3)$$

After predicting the cosine and sine value of the minutia orientation, we computes the loss function as Eq. (3). It should also be noted that while computing the gradient during the backpropagation process, the arccosine function may fail when the cosine value is 0 or 1 . Therefore, we clip the cosine value $(\cos\alpha\cdot\cos\beta+\sin\alpha\cdot\sin\beta)$ into $\left[10^{-6}, 1-10^{-6}\right]$ to avoid such possibility.

C. Ground Truth Minutiae Map Generation

The proposed algorithm is trained to generate the ground truth minutiae location heatmap. Therefore it can be challenging to accurately compute the ground truth minutiae map. In this context, we should note the following facts:

- There are only a few minutiae points or pixels as compared with the entire fingerprint image pixels. Therefore, if we only assign the minutiae pixels as positive and all other pixels as negative, the negative and positive ratio will be highly unbalanced.
- Minutiae are some key points in the fingerprint images, but they should not be considered as the accurate points as these can be slight differences (several pixels difference) between the ground truth minutiae labeled by two experts (or even minutiae marked by the same expert at different times) [31], [32]. Therefore, in the context of our application, the minutiae localization should better be regarded as a rough region rather than an accurate point.
- Some minutiae in a fingerprint image are quite close to each other. When there are only a few pixels distance between such minutiae, it is quite possible to detect only one minutia if the ground truth minutiae region circles are too large.

Reference [23] partitions a image into several 4×4 cells, then assigns a positive label to a cell if it contains a minutia and a negative label to a cell if its neighborhood does not include any minutia. The work detailed in [25] also generates a size reduced location map of size $\frac{H}{4}\times\frac{W}{4}$. It produces the ground truth location map using a Gaussian kernel and then computes element-wise maximum for the different minutiae. There are several limitations with both of these methods. The method in [23] attempts to balance the data by using the same number of positive and negative cells. However, since there are only a few minutiae in a fingerprint image, it discards most

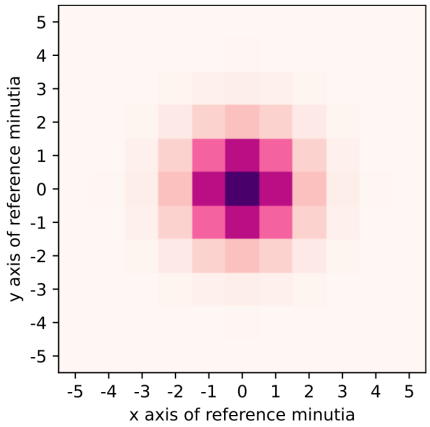


Fig. 5: Minutiae location map generated in [25], each pixel in this figure represents a patch with size of 4×4.

cells during the training and thereby significantly decreasing the training speed.

The biggest problem in [25] is that cells around the minutia localized cell will also be labeled with a probability between 0 and 1. Fig. 5 presents the location heatmap of [25], from where we can observe that the eight neighboring cells around the cell that contains the minutia (or the center cell) have a high probability, and the 5×5 cells around the center cell have probabilities that are greater than 0 . Every cell has a size of 4×4; therefore, 20×20 pixels around the minutiae are not labeled as negative, while 12×12 pixels around the minutiae are labeled with high probability. This is unresonable because one minutia should only be within a small region, and only pixels in this region should be labeled as positive. Especially for two minutiae that are close to each other, it is easy to misdetect only one minutia when training with such a location heatmap. Moreover, the location heatmap is not binary, and it cannot directly use the binary cross-entropy loss or the focal loss [33].

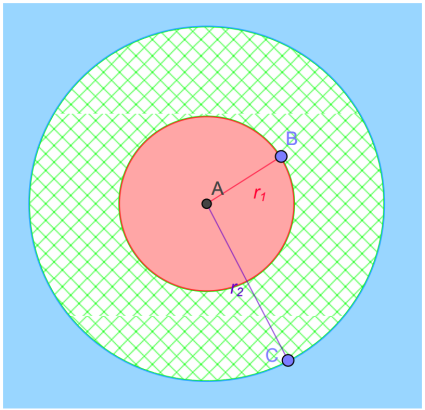


Fig. 6: Illustration for the generation of minutiae localization. The red region represents positive, blue region represents negative while the other green shadow region is not used. The r_1 and r_2 here represents the radius of the two circular regions.

Therefore, we propose a new method to address such limitations. Firstly, since a minutia is a region instead of an

accurate point, we use the labeled minutia location as the center and consider circular region with a small radius (r_1). Every pixel in this circle is regarded as positive. Next, as the minutia is located in a small region, we can consider another concentric circle with a greater radius (r_2). Pixels outside the second circle are at least r_2 pixels away from the manually labeled location and are regarded as negative. We can then discard pixels located in the cirque region and exclude them for the training because they are difficult to be classified positive or negative. In addition, as there are many minutiae, one pixel may have different labels for different minutiae. Therefore, we iterate every minutia, pixels marked as positive for at least one time is labeled as positive, pixels marked as "not used" for at least one time is labeled as "not used", and all other pixels are labeled as negative. Fig. 6 provides a visualization of this ground truth minutia location map generation method, where the pixels in the annulus region between the red positive region and blue negative region are discarded during the process of training.

D. Dynamic End-to-end Loss Design

In order to balance the positive and negative pixels, focal loss [33] is used to force the network to focus on positive pixels during the training process. Suppose $\widehat{p}$ is the predicted probability for one pixel and y is the ground truth label (either 1 or 0), then the loss can be written as follows:

$$FL = -y \times \alpha(1-p)^{\gamma}\log(p) - (1-y)\times(1-\alpha)p^{\gamma}\times\log(1-p). \tag{4}$$

The parameters α and γ are used to control the training, where α is used to balance the positive and negative ratio, and γ is used to control the training for easy-to-classify and hardto-classify examples. The higher is the value of γ, the lower is the loss for well-classified examples, making the network turns its attention to hard-to-classify examples. The total or final minutiae location map loss (L_l) is the mean value of all pixels' focal loss. The pixels labeled as "not used" are excluded in the computation.

In addition, after computing the minutiae orientation map, we recover the predicted orientation for every minutia, then compute the mean loss for the orientation (L_o) by using Eq. (3).

Finally, we combine these two losses to one loss and jointly train the network, such loss function can be defined as follows:

$$L = L_l + f_i \times L_o, \tag{5}$$

where parameter f_i is a function of current epoch number i and is used to dynamically train the network. We do not use a constant value here because then it is hard to optimize both two networks. At the beginning of the training process, the network should focus on training the minutiae location map; therefore, the parameter f_i should be small. After several epochs' of training, the network backbone can be well trained, and the parameter f_i should be increased to enable the network concentrate on optimizing the the loss for the orientation. Therefore, we formulate the function f_i as follow:

$$f_i = \min\left(\beta_{\text{init}} \times 10^{\lfloor \frac{i}{M} \rfloor}, \beta_{\max}\right). \tag{6}$$

where M, β_{init} , and $\beta_{\max}$ are three predefined parameters, L_o learning rate, f_i will be initialized as β_{init} and then increase 10 times every M epoch until it reaches to $\beta_{\max}$. By using such kind of loss combination, we dynamically and jointly, train the minutiae location and orientation network. The learning process is faster and can also ensure better results.

E. Minutiae Location Estimation

The proposed network predicts a minutiae location map, and we then need to locate the minutiae based on the location map. The easiest way of locating minutiae is to set a threshold, find all connected components which values are equal to or greater than this threshold, and compute the center of every connected component. This method is quite easy but does not work well. The main problem is that is is hard to set a proper threshold. When setting a small threshold, it may detect many imposter minutiae. Besides, it is also quite possible that some minutiae closely located are detected as one minutiae. In contrast, when setting a big threshold, it may fail to detect many genuine minutiae. Therefore, we propose a new method to address such problems, which is introduced as follows.

$$p = \sqrt{\text{area}} \cdot \left(e^{\tau - \tau_{\text{init}} + 0.1} - 1\right). \tag{7}$$

First, we use a small probability threshold (τ_{init}) and find all connected components ($C_1, C_2, \ldots, C_m$). Next, we increase the threshold and find the connected components in every component C_i recovered earlier. We continue this process until these components has less than a fixed component area threshold (τ_{area}) of pixels. If more than one connected components are found in the previous component, it means two or more minutiae conglutinate together, and we continue such process for every new connected component. Finally, we get many non-coincident connected components, and we compute the center location of every component to locate all possible minutiae. Besides, we also compute the possibility for every minutia based on the component area and corresponding threshold using Eq. (7). Here area is the number of pixels in every connected component, and τ is the current threshold. It may compute several different probabilities for one minutia, and we only use the maximum one. We use Eq. (7) to balance component area and its value. Every possible minutia's connected component in the minutiae location map is like a circle. The probability of this minutiae is proportional to the radius, or $\sqrt{\text{area}}$. Besides, the probability is also related to the value of corresponding connected components. As a result, Eq. (7) can well balance these two factors and compute a reliable probability score. Finally, we retrieve every possible minutia and corresponding score, and the closely located minutiae that are also expected to be accurately located.

F. Synthetic Fingerprint Generation

During the last few years, deep learning-based algorithms has been playing a more and more important role in fingerprint recognition area, which require large-scale fingerprint datasets

for training and evaluation. However, it has inherent risks and privacy concerns to collect and share large-scale fingerprint databases. For example, the National Insitute of Standards and Technology (NIST) has withdrawn several publicly available fingerprint databases due to privacy related issues [34], e.g., NIST Special Database 4, NIST Special Database 9, NIST Special Database 14 , et al. Generating synthetic fingerprint images can address both the data collection cost problem and privacy problem, as synthetic fingerprints can be generated easily within low cost and alleviate the need for the individuals identity of the subjects used during training.

Traditional fingerprint synthesis algorithms usually involves sampling from independent statistical models for orientation field and minutiae with Gabor-filtering or other models to generate the final ridge structure [35]-[37]. There are several shortcomings for these traditional approaches. Firstly, the minutiae distribution of generated fingerprint are very different from real fingerprints. Second, the independent modeling used in these traditional methods is not necessarily able to capture the correlation between the minutiae patterns, ridge valley structure and orientation field [26].

Therefore, many new GAN-based algorithms has been proposed recently. GANs can learn deep representations without using extensively annotated training data. They derive back-propagation signals through a competitive process involving a pair of networks [38]. It usually contains two networks, one network is used as an art forger and the other network is used as an art expert. The forger, known as the generator $(\mathcal{D})$ in the GAN literature, creates forgeries to make realistic images. The expert, known as the discriminator $(\mathcal{G})$ in the literature, receives both forgeries and real/authentic images, and aims to distinguish them. GANs don't rely on independent statistical models, thus it is suitable for synthetic fingerprint generation. Recently, GAN models have been employed for synthetic fingerprint generation [26], [39]-[41]. During the experiments, we adopt the synthetic fingerprint images generated using [26] as this method can generate high quality fake fingerprint images which are very similar to the real fingerprints.

III. EXPERIMENTS AND RESULTS

A. Databases and Protocols

In order to present comparative performance evaluation with state-of-the-art methods, we first perform experiments on four publicly available contact-based fingerprint databases: ZJU Database [42], FVC2002 [43] Database, FVC2004 [44] Database, and IIITD MOLF Database [45]. During the experiments, we use the synthetic fingerprint database generated by [26] for pre-training. In the following, we provide more detailed information on these databases and the corresponding protocols used in our experiments for the performance evaluation. We also performed additional experiments in another two contactless fingerprint databases: PolyU Cross Sensor Database [46] and Benchmark 2D/3D database [47]. The following provides more detailed information about all databases and responding protocols used on our experiments.

1) FVC2002 and FVC2004 Database: FVC2002 Database and FVC2004 Database have four sub-databases with 800 fingerprint images from 100 subjects in each sub-database $(8$ fingerprint images for each subject). For fair comparisons, we follow the same protocol in [23], and perform experiments on four sub-databases: FVC2002-DB1A, FVC2002-DB3A, FVC2004-DB1A, and FVC2004-DB3A. We randomly select half of the subjects in each sub-database for training and then used the other half for test. Similar to [23], we do not scale the size of any fingerprint images as all fingerprint images have the fixed dpi (500 dpi). In order to measure the performance, we also use the same protocol and compared our extracted minutiae with the ground truth minutiae manually labeled by experts [48]. Suppose one minutia is within 12 pixels of one ground truth minutia, and the orientation difference between these two minutiae is within 20 degrees. In that case, the predicted minutia is correct, or else it will be a wrong minutia. Finally, we compute the precision, recall and use the F1 score to measure the detection performance. Since the training and test database is randomly split, the databases split will significantly influence the final results. Therefore, we perform experiments thrice with different random seeds and then compute the mean $\mathrm{F1}$ score. In order to ensure full reproducibility, we also provide the exact training and test subjects number that we use in our experiments, along with the source code via [49]. In addition, we also attempted to firstly pre-train our model using synthetic fingerprint database, then fine-tuning on the train images, and evaluate the performance on the test images with the same protocol.

2) The IIIT-D Multi-sensor Database: The IIIT-D Multisensor Optical and Latent Fingerprint (MOLF) database [45] is a large database that has 19,200 fingerprints images acquired from 100 subjects using five different sensors. We used the second (DB2) and third (DB3) databases during our experiments because DB1 has high-quality images and are easy to detect, while DB4 and DB5 are low-quality latent fingerprint databases, and are hard to manually label the ground truth minutiae for training. Both $\mathrm{DB2}$ and $\mathrm{DB3}$ contain 4,000 fingerprint images acquired from 1,000 fingers of 100 persons, with four images for each finger. During our experiments, we use the first 200 fingerprint images from the first 5 subjects for training and the other 3800 images for the test. Similarly, we firstly detect the minutiae of training images by using VeriFinger [50] first and then manually inspect such detections and correct such labels when required. To evaluate the performance, we match the extracted minutiae by using MCC [4]. We also use challenging all-to-all protocol for more reliable performance estimation, therefore generating 5,700 genuine match scores and 7,992,300 imposter match scores.

3) The ZJU Fingerprint Database: The ZJU Finger Photo and Touch-based Fingerprint Database contains both 9,898 contactless and contact-based fingerprint images from 206 subjects. Each subject contains several images from two left hand fingers and two right hand fingers. All contact-based fingerprint images were acquired using a URU 4500 optical-based scanner at 512 ppi. All subjects are in this database have been partitioned into six groups, A, B, C, D, E and F, we select the first four subjects in every group as the training database and the other as the part of the test database. Therefore, it contains 1,153 training fingerprint images and

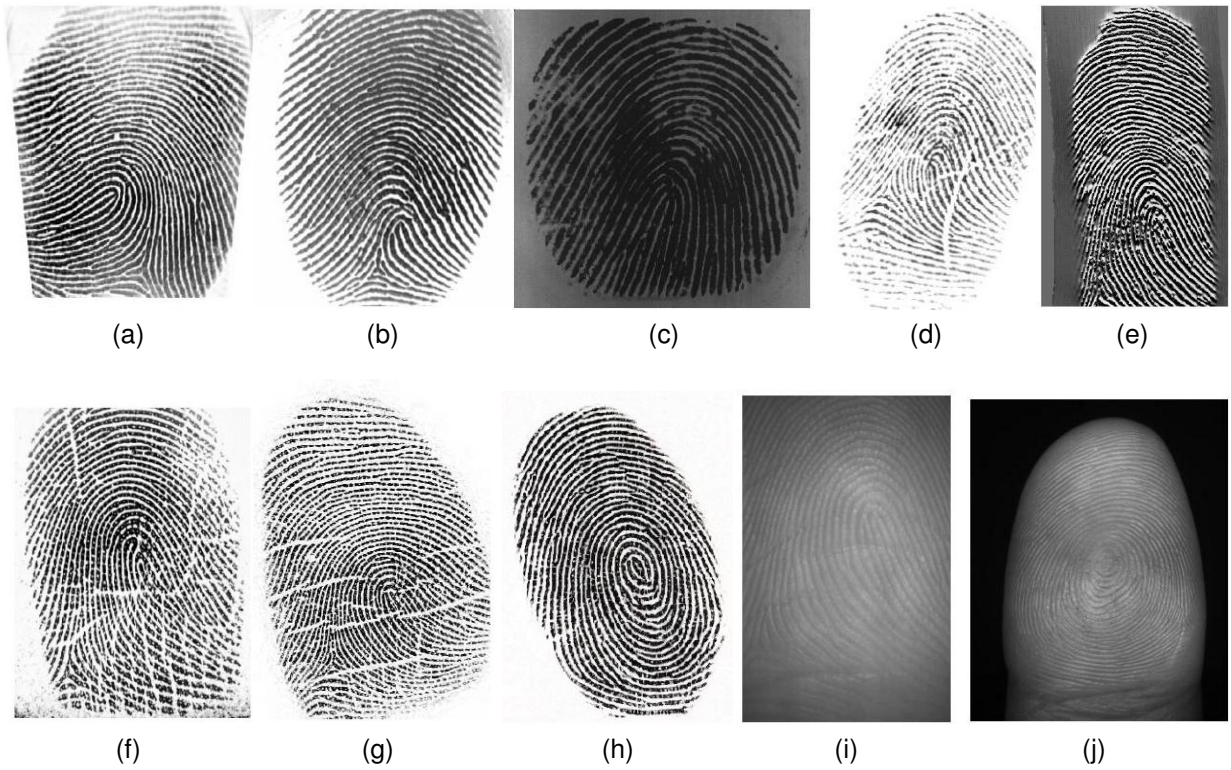


Fig. 7: Sample fingerprint images from different fingerprint databases used in this work. From left to right, the fingerprint images are from (a). ZJU database [42], (b). FVC2002-DB1A [43], (c). FVC2002-DB3A [43], (d). FVC2004-DB1A [44], (e). FVC2004-DB3A [44], (f). MOLF-DB2 [45], (g). MOLF-DB3 [45], (h). GAN-based Synthetic Fingerprint database, (i). PolyU-contactless Fingerprint Database [46], (j). Benchmark 2D/3D Fingerprint Database [47].

8,745 test images. We first pre-train the proposed model on the synthetic fingerprint database, then we fine-tune the model on the training fingerprint images, and finally test on the test part of this database. Since the numbers of images in the test part of this database are large, the number of match scores generated from our experiments are very large, i.e., includes 48,163 genuine scores and 43,139,019 imposter match scores. To evaluate the performance of the proposed algorithm, we present comparisons with VeriFinger [50] using exactly the same test database and the images.

4) Synthetic Fingerprint Database: The Clarkson Fingerprint Generator [26] can generate high-quality contact-based fingerprint images within little expanse. [26] provides the source code [51], along with a synthetically generated fingerprints database, which contains 50,048 synthesis fingerprint images. To benefit the reproduction of the proposed algorithm, we directly used the provided database and did not randomly generate fingerprint images. In order to generate ground truth minutiae for the network training, we firstly detect all the possible minutiae from the synthesized database [26] using the VeriFinger [50] and exclude those minutiae which have very low quality score. During our experiments, we pre-trained the deep learning model under this database and then test on several other contact-based fingerprint databases.

5) PolyU Cross Sensor Fingerprint Database and Benchmark 2D/3D Database:: PolyU Cross Sensor Database [46] is a two-session cross sensor fingerprint database. The first session contains 336 subjects with 6 contactless and contact-based fingerprint images for each subject, and the second session contains 160 subjects with 6 images for each subject. Benchmark 2D/3D [47] consists of 9000 contactless fingerprint images which are acquired from 1500 subjects. We train our network using the same protocol as in [25]. First, we select the contactless fingerprint images from fingers numbered between 1 and 136 in both PolyU Cross Sensor Database sessions as the training set (1440 contactless fingerprint images in total). Rest of the 1200 images from the other 200 subjects are used for performance evaluation. For the Benchmark 2D/3D database, we use the 400 contactless fingerprint images from the first 200 subjects for fine-tuning and then use the remaining 2600 images from 1300 subjects for the test. In order to efficiently obtain the ground truth minutiae of training images, we first use VeriFinger [50] to extract all the minutiae from training images in the datavase. Then we manually correct the location and orientation of several incorrectly detected minutiae using the popular LabelMe [52] software. Following the same protocol in [25], we use the existing Minutia CylinderCode (MCC) [4], [53] for matching the minutiae templates. We also perform both fingerprint verification using the same protocol as in [25]. For PolyU Cross database, we generate 3000 genuine and 19900 imposter match scores. For Benchmark 2D/3D

database, we generate 1300 genuine and 844,350 imposter match scores.

B. Additional Experimental Details

It is worth mentioning that we use the following data augmentation methods in our experiments: FlipBlur (including Gaussian Blur, Average Blur, Median Blur), Gamma Contrast, Gaussian Noise, Sharpen, Rotation. After data augmentation, we use adaptive histogram equalization to enhance the fingerprint images. During the test phase, we also use the same enhancement method.

There are several parameters that were empirically fixed as the same for all experiments in this paper. We use Adam optimizer during the training, the learning rate is set as 0.001 and the batch size is 16 . We empirically set the parameters α and γ in Sec. II-D as 0.9 and 2.5 respectively for all the experiments in this paper. Besides, during the experiment, we set r_1 as three and r_2 as six. The parameters τ_{init} and τ_{area} in Eq. (7) are set as 0.4 and 15 in our experiments respectively. All parameters can also be found in the source code [49].

C. Experimental Results on FVC Databases

We first perform experiments on FVC2002 and FVC2004 databases and then perform comparisons with [23]. Tab. I presents mean F1 scores from the proposed algorithm and [23] using the same or four databases. It is easy to observe that the proposed algorithm can achieve much higher accuracy over this baseline method.

TABLE I. Comparative results using FVC databases.

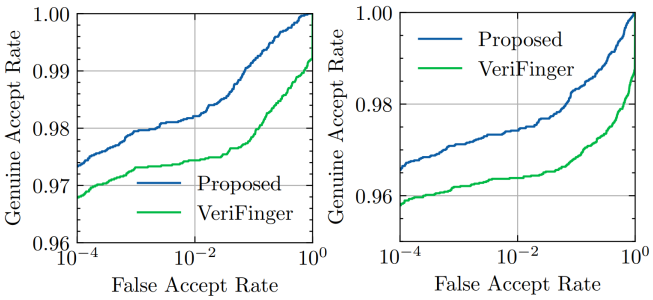
Database	Mean F1 [23]	Mean F1[1]	Mean F1[2]	Mean F1[3]
FVC02-DB1A	0.879	0.910	0.880	0.898
FVC02-DB3A	0.854	0.868	0.853	0.874
FVC04-DB1A	0.845	0.878	0.829	0.866
FVC04-DB3A	0.821	0.833	0.818	0.836
All	0.849	0.875	0.839	0.867

1) Results using the proposed algorithm.
2) Results using the proposed method (under stricter criterion).
3) Results pre-trained on the synthesis database and fine-tuned on the training database (under stricter criterion).

In order to ascertain the effectiveness of the proposed algorithm, except for using the same minutiae matching metric (location and orientation tolerance is 12 pixels and 20 degrees), we also use a stricter criterion and compute the F1 score, which is described as follow: One predicted minutia is correct if and only if there is a ground truth minutia located within 6 pixels distance from the expected minutia, and the orientation tolerance is within 10 degrees. As shown in Tab. I, the results under the stricter protocol are similar to the results from [23] under the easier protocol, which can further validate the effectiveness of the proposed algorithm in extracting the minutiae with high accuracy. In addition, we also pre-trained the proposed model and fine-tuned on the training database, which yields a better performance. Compared with results without pre-training, It achieves 0.867 mean F1 score under stricter criterion, compared with the result without pretraining (0.839), which evaluates the effectiveness of pretraining under synthesis databases.

TABLE II. Comparative results on IIITD MOLF database.

	Method	AUC (%)	EER (%)
DB2	NIST mindtct [54]	77.21	30.46
	VeriFinger [50]	98.68	2.51
	Proposed	99.69	1.65
DB3	NIST mindtct [54]	60.01	42.12
	VeriFinger [50]	97.73	3.47
	Proposed	99.18	2.33



(a) DB2 database (b) DB3 database

Fig. 8: Comparative experimental results on IIITD MOLF database.

D. Experimental Results on IIITD MOLF Database

In this section, we present experimental results on IIITD MOLF Database. We consider three baseline methods to extract the minutiae and resulting minutiae templates were matched by using MCC [4]. Tab. II provides the comparative summary from AUC and EER. These results consistently validate the effectiveness of the proposed method and also achieve superior accuracy than VeriFinger [50] which is widely considered [23], [25], [27], [46] as a very strong baseline, especially for the contact-based fingerprint images.

We also provide the ROC curves in Fig. 8. Due to relatively low performance from [54], we do not plot the ROC in Fig. 8. As discussed in Sec. III-A, we generate around six thousand genuine match scores and 8 million imposter match scores using both IIITD MOLF DB2 and DB3 database.

E. Experimental Results on ZJU Database

In this section, we present experimental results on ZJU Fingerprint Database. Similar to IIITD MOLF database, we consider two baseline methods to extract the minutiae and then match the minutiae templates using [4]. This dataset is of very large size, and most of the contact-based fingerprint images have relatively high quality; therefore both VeriFinger [50] and the proposed method can achieve extremely high accuracy under this database.

Fig. 9 compares the ROC curves VeriFinger achieves 0.69% EER, compared with 0.37% EER acquired using the proposed method. It should be noted that VeriFinger cannot detect the minutiae from 19 fingerprint images. In order to ensure a fair comparison, we consider the corresponding matching scores (both genuine and imposter) as zero.

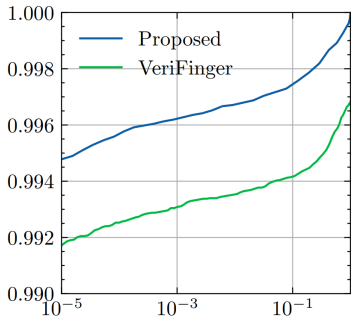


Fig. 9: Comparative ROC curves on ZJU database.

TABLE III. Comparative results on PolyU Cross Sensor Database.

Method	AUC (%)	EER (%)
NIST mindtct [54]	58.91	36.85
MinutiaeNet [22]	93.03	13.35
VeriFinger [50]	98.16	2.99
ContactlessMinuNet [25]	99.33	1.94
Proposed	99.25	1.90

F. Experimental Results on PolyU Database

We perform experiments on PolyU Cross Sensor Fingerprint Database, the comparative results are shown in Tab. III. From the table, it is easy to observe that the proposed algorithm outperforms state-of-the-art *contactless* fingerprint minutiae extraction methods.

G. Experimental Results on Benchmark Database

In this section, we present experimental results from the proposed method on Benchmark 2D/3D Database. We follow the same protocol as [25] to ensure fairness and summarize comparative results in Tab. IV.

It can be observed from above table III and IV that the proposed algorithm can offer much higher accuracy than state-of-theart methods.

IV. Discussion

A. Minutiae Map Feature Visualization

We performed Grad-CAM [55] based qualitative analysis of the minutiae map features extracted by the proposed model. Grad-CAM is a popular deep network based tool for visualizing where a convolutional neural network concentrating on, which can help us to gain better understanding of a model. Therefore, we present the the Grad-CAM visualization results from several test image samples using [56] in Fig. 10.

TABLE IV. Comparative results on Benchmark 2D/3D Database.

Method	AUC (%)	EER (%)
NIST mindtct [54]	81.84	4.28
MinutiaeNet [22]	79.74	26.34
VeriFinger [50]	95.44	9.02
ContactlessMinuNet [25]	98.24	4.28
Proposed	98.91	3.89

TABLE V. Comparative results with and without local MinNet.

Database	Mean F1 (original network)	Mean F1 (without loal MinNet)
FVC02-DB1A	0.898	0.803
FVC02-DB3A	0.874	0.780
FVC04-DB1A	0.866	0.782
FVC04-DB3A	0.836	0.761
All	0.867	0.785

TABLE VI. Comparative results for new loss function.

Database	Mean F1 (proposed loss)	Mean F1 ($\Delta\theta$)	Mean F1 (cosine loss)
FVC02-DB1A	0.898	0.888	0.865
FVC02-DB3A	0.874	0.864	0.847
FVC04-DB1A	0.866	0.855	0.839
FVC04-DB3A	0.836	0.829	0.810
All	0.867	0.856	0.841

From these images in Fig. 10, we can observe that the proposed model concentrates on the minutiae, and most minutiae in the sample test fingerprint images are correctly concentrated on, which underlines the effectiveness of the proposed model.

B. Ablation Study on the Effectiveness of Local MinNet

We examine the contributions from the local MinNet by eliminating it from the whole network. The local MinNet is used to generate a local minutiae map, and it can be directly removed from the network without influencing the whole network architecture. In order to provide more effective comparison, we trained the models with sufficient data. We trained the model on the synthetic database and then fine tuned on the FVC2002 and FVC2004 fingerprint database.

Tab. V presents the comparative results with and without local MinNet. From the table, we can easily find that without local MinNet, the accuracy decrease significantly after removing the local MinNet. This is mainly because that many detail features are lost after removing the local MinNet. The global MinNet can only generate a size reduced minutiae map, and many minutiae cannot be detected correctly.

We also compute the complexity of local MinNet. The total network has 2.49G FLOPs, while the local MinNet has 0.83G FLOPs, which occupies around $\frac{1}{3}$ of the whole network complexity. Therefore, using the local MinNet to improve the accuracy is not without any cost. It significantly increase the computational complexity because it takes the full size feature as the input of every convolutional layer.

C. Ablation Study on the Effectiveness of the New Loss Function for the Orientation Estimation

We examine the contributions from the proposed loss function. Tab. VI presents the comparative results using different loss function. From the table, we can easily note that compared with directly using cosine distance, it slightly improves the minutiae detection accuracy by using the proposed loss function. As the orientation is relatively easy to be detected and the matching metric is relatively loose (the orientation tolerance is

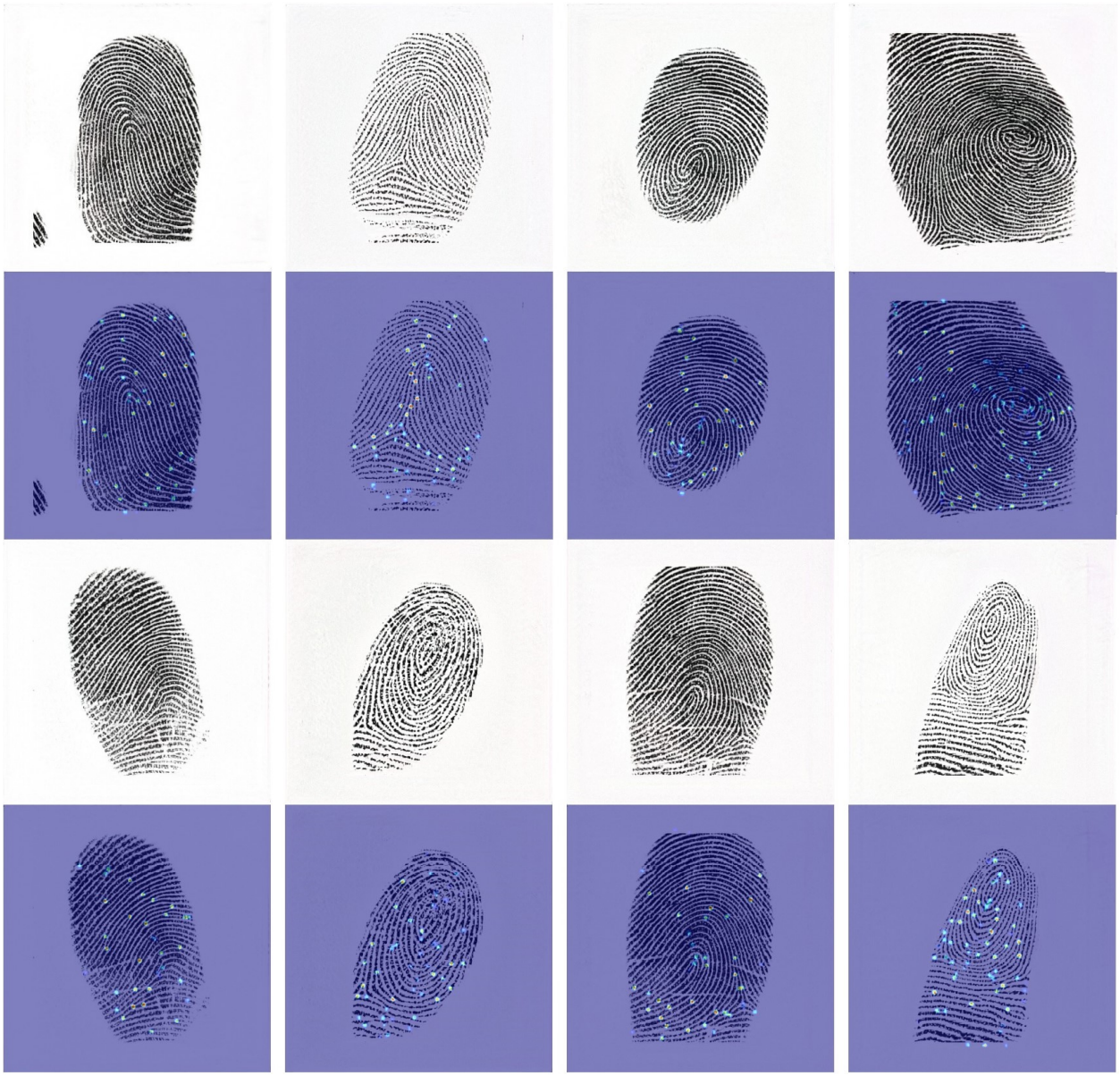


Fig. 10: Sample fingerprint images and their corresponding Grad-CAM [55] visualizations.

10 degrees), the improvement is not significant. Nevertheless, the comparative results still validate the effectiveness of the proposed loss function.

D. Ablation Study on the Effectiveness of the Ground Truth Minutiae Map Generation

We also examine the contributions from the ground truth minutiae map generator. We perform experiments using five different sets of r_1 and r_2, where r_2 is always 6 and r_1 is from 2 to 6 . It should be noted that when $r_1 = r_2 = 6$, all pixels are used for training and this minutiae map is a normal map. Tab. VII presents the comparative results using these five different r_1 and r_2. From the table, we can find that compared with using all pixels, it can yield better performance by discarding some pixels as the proposed way. The total mean F1 score increased from 0.834 to 0.867, which validate the effectiveness of the proposed ground truth minutiae map generation method.

E. Computational Complexity Analysis

We compare the computational complexity of several stateof-the-art deep learning-based minutiae detection algorithms and summarize these results in Tab. VIII. It can be observed from the table that the proposed algorithm is much faster than state-of-the-art deep learning-based minutiae detection algorithms in the literature. We can note that algorithm in [23] requires quite high FLOPs. In fact, after feature extraction, [23] generates many (100 in their experiments) $7 \times 7 \times 256$ candidates and uses a fully connected network to predict the position, orientation and the probability. It's computational requirement is 12.84M FLOPS for only one minutiae prediction, and therefore it requires 1.28G FLOPs

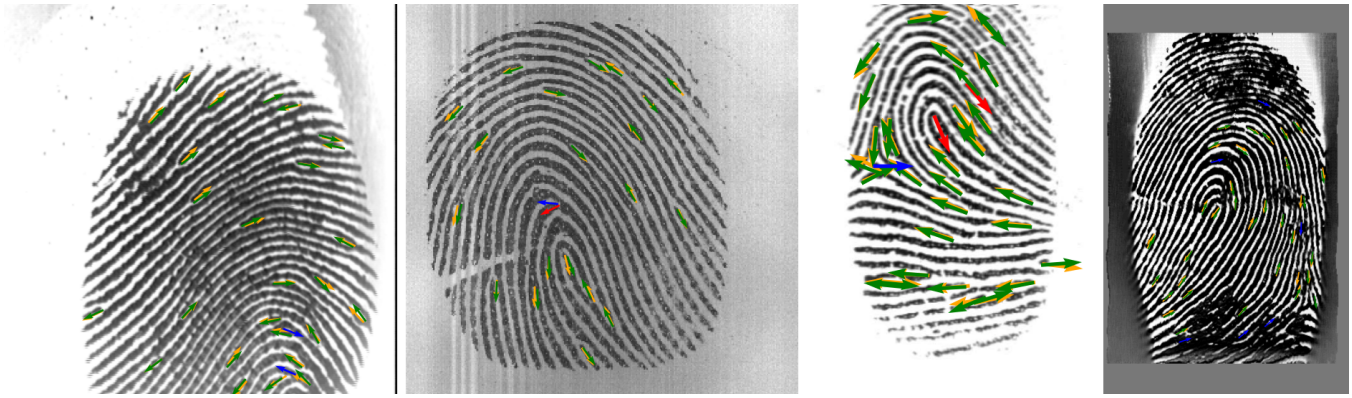


Fig. 11: Minutiae extraction results samples, green arrows are minutiae marked by experts, yellow arrows are correct minutiae predicted by proposed method, red arrows are wrong predicted minutiae, and blue arrows are manually labeled minutiae. From left to right, the fingerprint images are from FVC2002 DB1A, DB3A, FVC2004 DB1A and DB3A respectively.

TABLE VII. Comparative results using different r_1 and r_2, the two integer numbers in the bracket are r_1 and r_2 respectively.

Database	Mean F1 (2, 6)	Mean F1 (3, 6)	Mean F1 (4, 6)	Mean F1 (5, 6)	Mean F1 (6, 6)
FVC02-DB1A	0.845	0.898	0.883	0.876	0.864
FVC02-DB3A	0.826	0.874	0.865	0.859	0.850
FVC04-DB1A	0.803	0.866	0.848	0.834	0.826
FVC04-DB3A	0.783	0.836	0.821	0.812	0.803
All	0.812	0.867	0.850	0.842	0.834

TABLE VIII. Comparative computational complexity for different deep learning-based algorithms.

Method	Parameters	FLOPs
FastMinutiaeExtract [23]	70.4M	26.6G
ContactlessMinuNet [25]	36.5M	5.45G
Proposed	2.2M	2.5G

for the entire 100 candidates. It also requires much more additional computations for the feature extraction.

Due to the novel designs introduced in Sec. II, the proposed algorithm has a lightweight but powerful network architecture, and it directly computes the minutiae location and orientation, instead of selecting many candidates and then performing the second stage prediction. Therefore, it is much faster as compared with state-of-the-art deep-learning based methods.

F. Minutiae Extraction Results Samples

We randomly select some fingerprint images from FVC2002 and FVC2004 databases, and visualize the extracted minutiae. Fig. 11 illustrates some sample fingerprint images with the minutiae extracted using the proposed algorithm. We can observe that most of the minutiae are accurate. The location and orientation of these correctly detected minutiae are quite close to the ground truth minutiae that are manually labeled by the experts. It is worth noting that there is only one misdetected minutiae in the second image sample in Fig. 11. The reason for such misdetection is that this minutiae as a kind of like a ridge ending, therefore the proposed algorithm detects a wrong orientation.

G. Additional Discussion on Computational Complexity

In Tab. IX, we present the average running time comparisons from several state-of-the-art deep learning-based minutiae detection algorithms in the literature. It will be unfair to directly compare the running time as the experiments are under different environments. Therefore, we perform the experiments using the same dataset as used in [22] and [23] under a more simplified computational hardware. We perform the experiments on a laptop computer with a GTX1050-Ti(Mobile) GPU card, which has much lower effective speed than the GPU (GTX1060 and TITAN XP) used in [22] and [23]. It can be easy to note from the results in Tab. IX that even we use a more simplified GPU card, our algorithm is still much faster than the state-of-theart deep learning-based algorithms, which can further validate the achievable speed from the proposed algorithm. Besides, we can note from Tab. IX that the proposed algorithm has a very different running time on FVC2004 [44] and Benchmark 2D/3D [47], which is mainly because that the image sizes of these two databases are different.

V. Conclusions and Further Work

In this paper, we proposed an end-to-end fingerprint minutiae extraction method. Unlike currently available state-of-theart algorithms, the proposed method uses a shallow pixel-wise local dilated neural network to extract the local features and a deep patch-wise network to extract the global features, then adaptively consolidates these two features to generate a pixelwise minutiae location and orientation map. In addition, we also introduce a new loss function for the orientation estimation and use a new dynamic end-to-end loss to jointly train the minutiae location and orientation networks. Another novel advancement is that we introduce a new method to automatically discard some pixels during the training process to improve the accuracy and learning speed. We extensively evaluated our algorithm using several contact-based and contactless fingerprint databases. The experimental results demonstrate that the proposed method outperforms state-of-the-art algorithms, with a much faster speed, and validate its effectiveness for the realworld applications. Despite attractive

TABLE IX. Comparative average running time of several state-of-the-art deep learning-based minutiae extraction methods. The results of proposed algorithm are computed by ourselves, while the other results are from reference [22] and [23].

Method	Environment (GPU)	Test database	Time(s)
MinutiaeNet([22])	GTX1060	FVC2004	1.2([23])
FingerNet([19])	GTX1060	FVC2004	0.32([23])
FME([23])	GTX1060	FVC2004	0.03([23])
Proposed	GTX1050-Ti(Mobile)	FVC2004	0.025
Multi-task FCN([25])	TITAN Xp	Benchmark 2D/3D	0.86([25])
MinutiaeNet([22])	TITAN Xp	Benchmark 2D/3D	1.2([25])
Proposed	GTX1050-Ti(Mobile)	Benchmark 2D/3D	0.083

minutiae detection performance and speed, more work needs to be done to realize a high-speed universal minutiae detector. It will be useful to evaluate the performance on more challenging fingerprint images, e.g. latent fingerprint images, and to extend this detector for detecting 3D fingerprint minutiae [1] and is part of further work in this area.

REFERENCES

[1] A. Kumar, Contactless 3D Fingerprint Identification. Springer International Publishing, 2018.

[2] D. Maltoni, D. Maio, A. K. Jain, and J. Feng, Handbook of fingerprint recognition. Springer Science & Business Media, 2022.

[3] Dingrui Wan and Jie Zhou, "Fingerprint Recognition Using ModelBased Density Map," IEEE Transactions on Image Processing, vol. 15, no. 6, pp. 1690-1696, Jun. 2006.

[4] R. Cappelli, M. Ferrara, and D. Maltoni, "MCC Software Development Kit (SDK) - Version 2.0," http://biolab.csr.unibo.it/research.asp, 2015.

[5] M. Ferrara, D. Maltoni, and R. Cappelli, "Noninvertible Minutia Cylinder-Code Representation," IEEE Transactions on Information Forensics and Security, vol. 7, no. 6, pp. 1727-1737, Dec. 2012.

[6] R. Cappelli, M. Ferrara, and D. Maltoni, "Fingerprint Indexing Based on Minutia Cylinder-Code," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp. 1051-1057, May 2011.

[7] K. Cao and A. K. Jain, "Automated Latent Fingerprint Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 4, pp. 788-800, Apr. 2019.

[8] X. Yin, Y. Zhu, and J. Hu, "Contactless Fingerprint Recognition Based on Global Minutia Topology and Loose Genetic Algorithm," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2841, 2020.

[9] A. A. Paulino, J. Feng, and A. K. Jain, "Latent Fingerprint Matching Using Descriptor-based Hough Transform," IEEE Transactions on Information Forensics and Security, vol. 8, no. 1, pp. 31-45, 2012.

[10] A. Farina, Z. M. Kovács-Vajna, and A. Leone, "Fingerprint Minutiae Extraction from Skeletonized Binary Images," Pattern Recognition, vol. 32 , no. 5 , pp. 877 − 889, 1999.

[11] Xiao Yang, Jianjiang Feng, and Jie Zhou, "Localized Dictionaries Based Orientation Field Estimation for Latent Fingerprints," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 5, pp. 955969, May 2014.

[12] X. Jiang, W.-Y. Yau, and W. Ser, "Detecting the Fingerprint Minutiae by Adaptive Tracing the Gray-level Ridge," Pattern recognition, vol. 34, no. 5, pp. 999-1013, 2001.

[13] J. Liu, Z. Huang, and K. L. Chan, "Direct Minutiae Extraction from Gray-level Fingerprint Image by Relationship Examination," in Proceedings of International Conference on Image Processing, vol. 2, 2000, pp. 427-430.

[14] A. Sankaran, P. Pandey, M. Vatsa, and R. Singh, "On Latent Fingerprint Minutiae Extraction Using Stacked Denoising Sparse AutoEncoders," in IEEE International Joint Conference on Biometrics, 2014, pp. 1-7.

[15] A. Malhotra, A. Sankaran, M. Vatsa, and R. Singh, "On Matching Finger-selfies using Deep Scattering Networks," IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 2, no. 4, pp. 350-362, 2020.

[16] L. Sifre and S. Mallat, "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 12331240.

[17] A. Chowdhury, S. Kirchgasser, A. Uhl, and A. Ross, "Can a CNN Automatically Learn the Significance of Minutiae Points for Fingerprint Matching?" in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 351-359.

[18] L. N. Darlow and B. Rosman, "Fingerprint Minutiae Extraction Using Deep Learning," in International Joint Conference on Biometrics, 2017, pp. 22-30.

[19] Y. Tang, F. Gao, J. Feng, and Y. Liu, "FingerNet: An Unified Deep Network for Fingerprint Minutiae Extraction," in International Joint Conference on Biometrics, 2017, pp. 108-116.

[20] H. Tan and A. Kumar, "Towards More Accurate Contactless Fingerprint Minutiae Extraction and Pose-Invariant Matching," IEEE Transactions on Information Forensics and Security, pp. 1-1, 2020.

[21] L. Jiang, T. Zhao, C. Bai, A. Yong, and M. Wu, "A Direct Fingerprint Minutiae Extraction Approach Based on Convolutional Neural Networks," in International Joint Conference on Neural Networks, 2016, pp. 571-578.

[22] D.-L. Nguyen, K. Cao, and A. K. Jain, "Robust Minutiae Extractor: Integrating Deep Networks and Fingerprint Domain Knowledge," in IEEE International Joint Conference on Biometrics, 2018, pp. 9-16.

[23] B. Zhou, C. Han, Y. Liu, T. Guo, and J. Qin, "Fast Minutiae Extractor Using Neural Network," Pattern Recognition, vol. 103, p. 107273, Jul. 2020.

[24] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961-2969.

[25] Z. Zhang, S. Liu, and M. Liu, "A Multi-Task Fully Deep Convolutional Neural Network for Contactless Fingerprint Minutiae Extraction," Pattern Recognition, vol. 120, p. 108189, 2021.

[26] K. Bahmani, R. Plesh, P. Johnson, S. Schuckers, and T. Swyka, "High fidelity fingerprint generation: Quality, uniqueness, and privacy," in 2021 IEEE International Conference on Image Processing (ICIP). IEEE, 2021, pp. 3018-3022.

[27] J. J. Engelsma, K. Cao, and A. K. Jain, "Learning a Fixed-length Fingerprint Representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.

[28] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video SuperResolution Using an Efficient Sub-Pixel Convolutional Neural Network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874-1883.

[29] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," arXiv:1511.07122 [cs], Nov. 2015.

[30] 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, Jun. 2016.

[31] A. Malhotra, A. Sankaran, M. Vatsa, R. Singh, K. B. Morris, and A. Noore, "Understanding ACE-V Latent Fingerprint Examination Process via Eye-Gaze Analysis," IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 3, no. 1, pp. 44-58, 2020.

[32] P. Grother, M. McCabe, C. Watson, M. Indovina, W. Salamon, P. Flanagan, E. Tabassi, E. Newton, and C. Wilson, "Performance and interoperability of the incits 378 fingerprint template," National Institute of Standards and Technology, Alghero, 2006.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980-2988.

[34] "Nist special database catalog," https://www.nist.gov/srd/shop/ special-database-catalog, 2022-06-01.

[35] R. Cappelli, D. Maio, and D. Maltoni, "Sfinge: an approach to synthetic fingerprint generation," in International Workshop on Biometric Technologies (BT2004), 2004, pp. 147-154.

[36] Q. Zhao, A. K. Jain, N. G. Paulter, and M. Taylor, "Fingerprint image synthesis based on statistical feature models," in 2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS). IEEE, 2012, pp. 23-30.
[37] P. Johnson, F. Hua, and S. Schuckers, "Texture modeling for synthetic fingerprint generation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 154159.
[38] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," IEEE signal processing magazine, vol. 35, no. 1, pp. 53-65, 2018.
[39] P. Bontrager, A. Roy, J. Togelius, N. Memon, and A. Ross, "Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution," in 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS). IEEE, 2018, pp. 1-9.
[40] K. Cao and A. Jain, "Fingerprint synthesis: Evaluating fingerprint search at scale," in 2018 International Conference on Biometrics (ICB). IEEE, 2018, pp. 31-38.
[41] S. Minaee and A. Abdolrashidi, "Finger-gan: Generating realistic fingerprint images using connectivity imposed gan," arXiv preprint arXiv:1812.10482, 2018.
[42] S. A. Grosz, J. J. Engelsma, E. Liu, and A. K. Jain, "C2cl: Contact to contactless fingerprint matching," IEEE Transactions on Information Forensics and Security, vol. 17, pp. 196-210, 2021.
[43] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2002: Second Fingerprint Verification Competition," in Object recognition supported by user interaction for service robots, vol. 3 . IEEE, 2002, pp. 811-814.
[44] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain, "Performance Evaluation of Fingerprint Verification Systems," IEEE transactions on pattern analysis and machine intelligence, vol. 28, no. 1, pp. 3-18, 2005.
[45] A. Sankaran, M. Vatsa, and R. Singh, "Multisensor Optical and Latent Fingerprint Database," IEEE access, vol. 3, pp. 653-665, 2015.
[46] C. Lin and A. Kumar, "Matching Contactless and Contact-Based Conventional Fingerprint Images for Biometrics Identification," IEEE Transactions on Image Processing, vol. 27, no. 4, pp. 2008-2021, 2018.
[47] W. Zhou, J. Hu, I. Petersen, S. Wang, and M. Bennamoun, "A Benchmark 3D Fingerprint Database," in International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2014, pp. 935-940.
[48] A. Mikaelyan and J. Bigun, "Ground Truth and Evaluation for Latent Fingerprint Matching," in Proceeds of IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2012, pp. 83-88.
[49] "Source code of the proposed algorithm," https://www.comp.polyu.edu.hk/ csajaykr/DLPG.html, 2022.
[50] "Neurotechnology Verifinger SDK - Version 12.1," https://www. neurotechnology.com/verifinger.html, Nov 2021.
[51] "Clarkson fingerprint generator," https://github.com/keivanB/Clarkson_Finger_Gen, 2021.
[52] K. Wada, "labelme: Image Polygonal Annotation with Python," https://github.com/wkentaro/labelme, 2016.
[53] R. Cappelli, M. Ferrara, and D. Maltoni, "Minutia Cylinder-code: A New Representation and Matching Technique for Fingerprint Recognition," IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 12, pp. 2128-2141, 2010.
[54] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. McCabe, S. Janet, and K. Ko, "User's Guide to NIST Biometric Image Software," Tech. Rep., 2007.
[55] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 618-626.
[56] J. Gildenblat and contributors, "Pytorch library for cam methods," https://github.com/jacobgil/pytorch-grad-cam, 2021.