End-to-End Latent Fingerprint Search

Kai Cao, *Member, IEEE,* Dinh-Luan Nguyen, *Student Member, IEEE,* Cori Tymoszek, *Student Member, IEEE,* and Anil K. Jain, *Fellow, IEEE*

Abstract—Latent fingerprints are one of the most important and widely used sources of evidence in law enforcement and forensic agencies. Yet the performance of the state-of-the-art latent recognition systems is far from satisfactory, and they often require manual markups to boost the latent search performance. Further, the COTS systems are proprietary and do not output the true comparison scores between a latent and reference prints to conduct quantitative evidential analysis. We present an end-to-end latent fingerprint search system, including automated region of interest (ROI) cropping, latent image preprocessing, feature extraction, feature comparison , and outputs a candidate list. Two separate minutiae extraction models provide complementary minutiae templates. To compensate for the small number of minutiae in small area and poor quality latents, a virtual minutiae set is generated to construct a texture template. A 96-dimensional descriptor is extracted for each minutia from its neighborhood. For computational efficiency, the descriptor length for virtual minutiae is further reduced to 16 using product quantization. Our end-to-end system is evaluated on three latent databases: NIST SD27 (258 latents); MSP (1,200 latents), WVU (449 latents) and N2N (10,000 latents) against a background set of 100K rolled prints, which includes the true rolled mates of the latents with rank-1 retrieval rates of 65.7%, 69.4%, 65.5%, and 7.6% respectively. A multi-core solution implemented on 24 cores obtains 1ms per latent to rolled comparison.

Index Terms—Latent fingerprint recognition, end-to-end system, deep learning, autoencoder, minutiae descriptor, texture template, reference fingerprint.

1 Introduction

LATENT fingerprints[1] are arguably the most important forensic evidence that has been in use since 1893 [1]. Hence, it is not surprising that fingerprint evidence at crime scenes is often regarded as ironclad. This effect is compounded by the depiction of fingerprint evidence in media in solving high profile crimes. For example, in the 2008 film The Dark Knight[2] a shattered bullet is found at a crime scene. The protagonists create a digital reconstruction of the bullet's fragments, upon which a good quality fingermark is found, unaffected by heat or friction from the firing of the gun, nor by the subsequent impact. A match is quickly found in a fingerprint database, and the suspect's identity is revealed!

The above scenario, unfortunately, would likely have a much less satisfying outcome in the real forensic case work. While processing of fingermarks has improved considerably due to advances in forensics, the problem of identifying latents, whether by forensic experts or automated systems, is far from solved. The primary difficulty in the analysis and identification of latent fingerprints is their poor quality (See Fig. 1). Compared to rolled and slap prints (also called reference prints or exemplar prints), which are acquired under supervision, latent prints are lifted after being unintentionally deposited by a subject, e.g., at crime scenes, typically resulting in poor quality in terms of ridge clarity and presence of large background noise. In essence, latent prints are partial prints, containing only a small section of the complete fingerprint ridge pattern. And unlike reference prints, investigators do not have the luxury of requesting a second impression from the culprit if the latent is found to be of extremely poor quality.

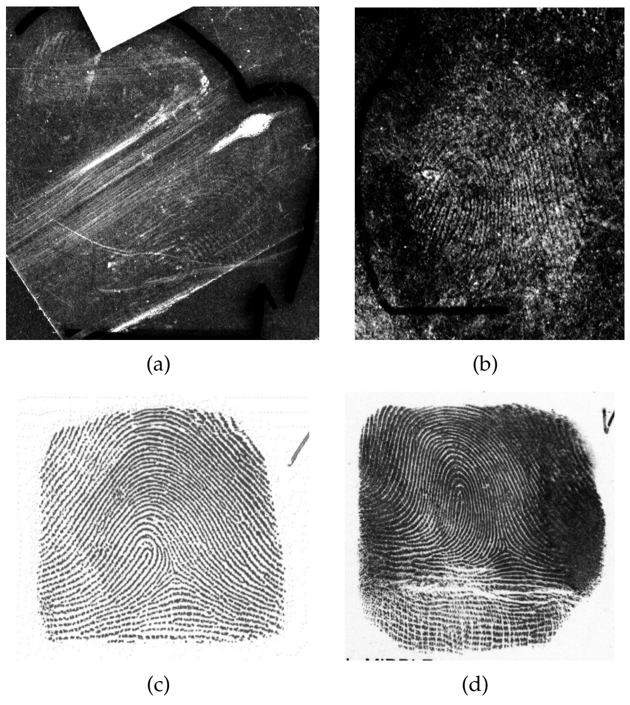


Fig. 1: Examples of low quality latents from the MSP latent database ((a) and (b)) and their true mates ((c) and (d)).

The significance of research on latent identification is evident from the volume of latent fingerprints processed annually by publicly funded crime labs in the United States. A total of 270,000 latent prints were received by forensic

• *Kai Cao, Dinh-Luan Nguyen, Cori Tymoszek and A.K. Jain are with the Dept. of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 U.S.A.*
E-mail: {kaicao,jain}@cse.msu.edu

1. Latent fingerprints are also known as latents or fingermarks
2. https://www.imdb.com/title/tt5281134/

labs for processing in 2009 [2] which rose to 295,000 in 2014, an increase of 9.2% [2]. In June 2018, the FBI's Next Generation Identification (NGI) System received 19,766 requests for Latent Friction Ridge Feature Search (features need to be marked by an examiner) and 5,692 requests for Latent Friction Ridge Image Search (features are automatically extracted by IAFIS) [3]. These numbers represent an increase of 6.8% and 25.8%, respectively, over June 2017 [3]. Every year, the Criminal Justice Information Services (CJIS) Division gives its *Latent Hit of the Year Award* to latent print examiners and/or law enforcement officers who solve a major violent crime using the Bureau's Integrated Automated Fingerprint Identification System, or IAFIS[3].

National Institute of Standards & Technology (NIST) periodically conducts technology evaluations of fingerprint recognition algorithms, both for rolled (or slap) and latent prints. In NIST's most recent evaluation of rolled and slap prints, FpVTE 2012, the best performing AFIS achieved a false negative identification rate (FNIR) of 1.9% for single index fingers, at a false positive identification rate (FPIR) of 0.1% using 30,000 search subjects (10,000 subjects with mates and 20,000 subjects with no mates) [4]. For latent prints, the most recent evaluation is the NIST ELFT-EFS where the best performing automated latent recognition system could only achieve a rank-1 identification rate of 67.2% in searching 1,114 latents against a background containing 100,000 reference prints [4]. The rank-1 identification rate of the best performing latent AFIS was improved from 67.2% to 70.2%[4] [5] when feature markup by a latent expert was also input, in addition to the latent images, to the AFIS. This gap between reference and latent fingerprint recognition capabilities is primarily due to the poor quality of friction ridges in latent prints (See Fig. 1). This underscores the need for developing automated latent recognition with both high speed and accuracy[5]. An automated latent recognition system will also assist in developing quantitative assessment of validity and reliability measures[6] for latent fingerprint evidence as highlighted in the 2016 PCAST [6] and the 2009 NRC [7] reports.

In the biometrics literature, the first paper on latent recognition was published by Jain *et al.* [8] in 2008 by using manually marked minutiae, region of interest (ROI) and ridge flow. Later, Jain and Feng [9] improved the identification accuracy by using manually marked extended latent features, including ROI, minutiae, ridge flow, ridge spacing and skeleton. However, marking these extended features in poor quality latents is very time-consuming and might not be feasible. Hence, the follow-up studies focused on increasing the degree of automation, i.e., reduction in the numbers of manually marked features for matching, for example, automated ROI cropping [10], [11], [12], [13], ridge flow estimation [12], [14], [15], [16] and ridge enhancement [17], [18], [19], deep learning based minutiae extraction [20], [21], [22], [23], and comparison [24]. However, these studies only focus on specific modules in a latent AFIS and do not build an end-to-end system.

Cao and Jain [25] proposed an automated latent recognition system which includes automated steps of ridge flow and ridge spacing estimation, minutiae extraction, minutiae descriptor extraction, texture template (also called virtual minutiae template) generation and graph-based matching, and achieved the state-of-the-art accuracies on two latent databases, i.e., NIST SD27 and WVU latent databases. However, their study has the following limitations: (i) manually marked ROI is needed, (ii) skeleton-based minutiae extraction used in [25] introduces a large number of spurious minutiae, and (iii) a large texture template size (1.4MB) makes latent-to-reference comparison extremely slow. Cao and Jain [26] improved both identification accuracy and search speed of texture templates by (i) reducing the template size, (ii) efficient graph matching, and (iii) implementing the matching code in C++. In this paper, we build a fully automated end-to-end system, and improve the search accuracy and computational efficiency of the system. We report results on three different latent fingerprint databases, i.e., NIST SD27, MSP and WVU, against a 100K background of reference prints.

2 Contributions

The design and prototype of the proposed latent fingerprint search system is a substantially improved version of the work in [25]. Fig. 2 shows the overall flowchart of the proposed system. The main contributions of this paper are as follows:

- An autoencoder based latent fingerprint enhancement for robust and accurate extraction of ROI, ridge flow and ridge spacing.
- An autoencoder based latent minutiae detection.
- Complementary templates: three minutiae templates and one texture template. These templates were selected from a large set of candidate templates to achieve the best recognition accuracy.
- Reducing descriptor length of minutiae template and texture template using non-linear mapping [27]. Descriptor for reference texture template is further reduced using product quantization for computational efficiency.
- Latent search results on NIST SD27, MSP, and WVU latent databases against a background of 100K rolled prints show the state-of-the-art performance.
- A multi-core solution implemented on Intel(R) Xeon(R) CPU E5-2680 v3@2.50GHz takes $\sim$1ms per latent-to-reference comparison. Hence, a latent search against 100K reference prints can be completed in 100 seconds. Latent feature extraction time is $\sim$15 seconds on a machine with Intel(R) i7-7780@4.00GHz (CPU) and GTX 1080 Ti (GPU).

3 Latent Preprocessing

3.1 Latent Enhancement via Autoencoder

We present a convolutional autoencoder for latent enhancement. The enhanced images are required to find robust and

3. https://www.fbi.gov/video-repository/newss-latent-hit-of-the-year-program-overview/view.

4. The best accuracy using both markup and image is 71.4% @ rank-1.

5. Automated latent recognition is also referred to as *lights-out recognition*; objective is to minimize the role of latent examiners in latent recognition.

6. Commercial AFIS neither provide extracted latent features nor the true comparison scores. Instead, only truncated and/or modified scores are reported.

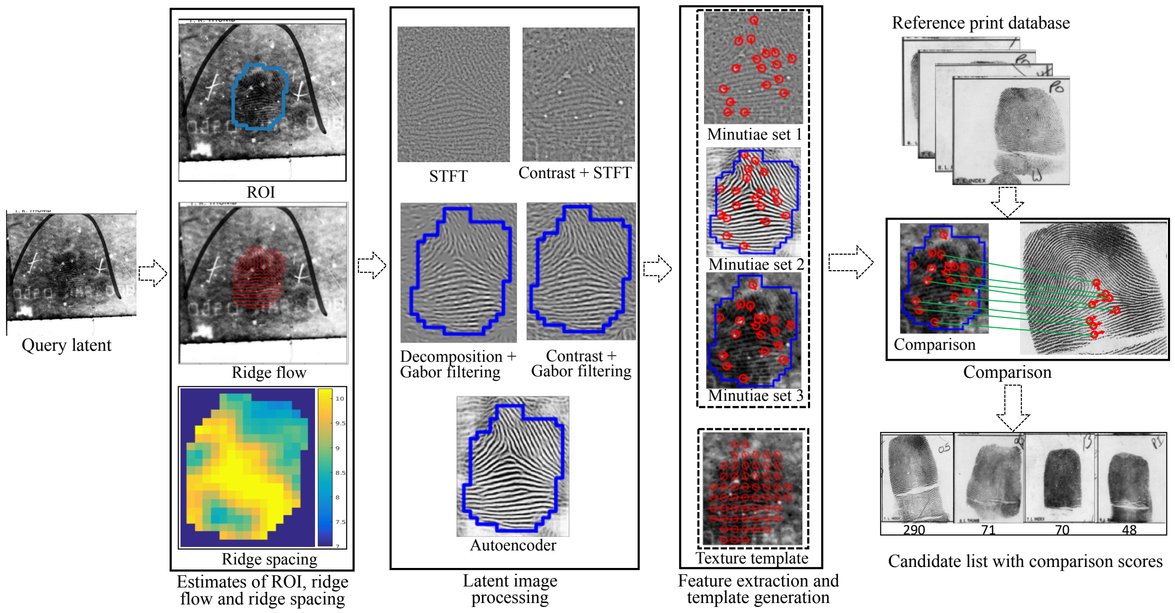


Fig. 2: Overview of the proposed end-to-end latent identification system. Given a query latent, three minutiae templates and one texture template are generated. Two matchers, i.e., minutiae template matcher and texture (virtual minutiae) template matcher are used for comparison between the query latent and reference prints.

accurate estimation of ridge quality, flow, and spacing. The flowchart for network training is shown in Fig. 3.

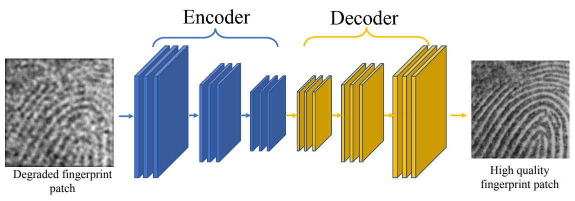


Fig. 3: A convolutional autoencoder for latent enhancement.

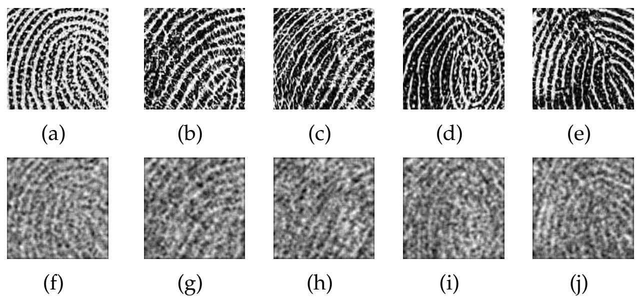


Fig. 4: Fingerprint patch pairs (128× 128 pixels) consisting of high quality patches (top row) and their corresponding degraded patches (bottom row) for training the autoencoder.

Since there is no publicly available dataset consisting of pairs of low quality and high quality fingerprint image for training the autoencoder, we degrade 2,000 high quality rolled fingerprint images (NFIQ 2.0[7] value > 70) to create image pairs for training. The degradation process involves randomly dividing fingerprint images into overlapping patches of size 128×128 pixels, followed by additive gaussian noise and Gaussian filtering with a parameter σ ($\sigma \in (5, 15)$). Fig. 4 shows some examples of high quality fingerprint patches and their corresponding degraded versions. In addition, data augmentation methods (random rotation, random brightness and change in contrast) were used to improve the robustness of the trained autoencoder.

7. NFIQ 2.0 [28] ranges from 0 to 100, with 0 indicating the lowest quality and 100 indicating the highest quality fingerprint.

The convolutional autoencoder includes an encoder and a decoder, as shown in Fig. 3. The encoder consists of 5 convolutional layers with a kernel size of 4×4 and stride size of 2, while the decoder consists of 5 deconvolutional layers (or transposed convolutional layer [29]) also with a kernel size of 4×4 and stride size of 2. The activation function ReLU (Rectified Linear Units) is used after each convolutional layer or deconvolutional layer with the exception of the last output layer, where the *tanh* function is used. Table 1 summarizes the architecture of the convolutional Autoencoder.

The autoencoder trained on rolled prints does not work very well in enhancing latent fingerprints. So, instead of raw latent images, we input only the texture component of the latent by image decomposition [12] to the autoencoder. Fig. 5 (b) shows the enhanced latent corresponding to the latent image in Fig. 5 (a). The enhanced latents have significantly higher ridge clarity than input latent images.

3.2 Estimation of Ridge Quality, Ridge Flow and Ridge Spacing

The dictionary based approach proposed in [12] is modified as follows. Instead of learning the ridge structure dictionary using high quality fingerprint patches, we construct the dictionary elements with different ridge orientations and spacings using the approach described in [30]. Fig. 6 illus-

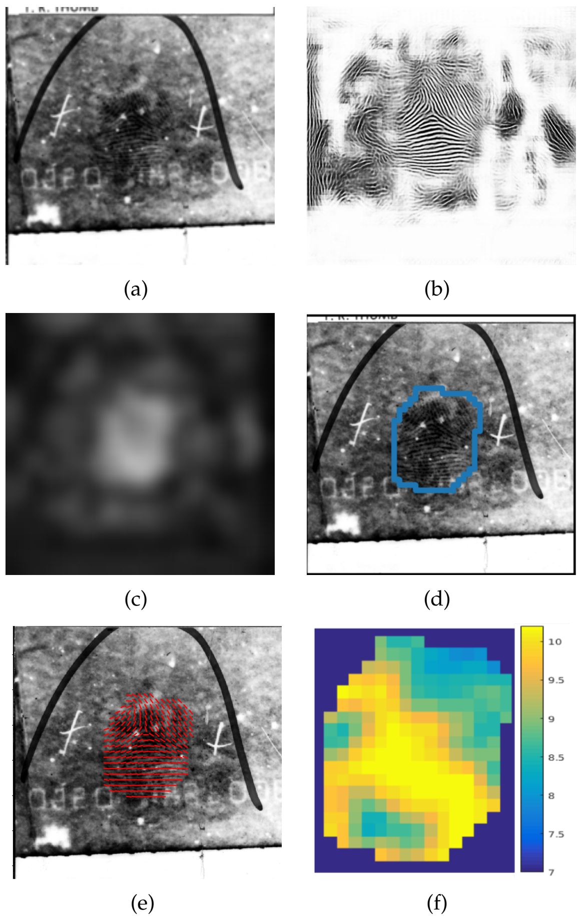


Fig. 5: Ridge quality, ridge flow and ridge spacing estimation. (a) Latent fingerprint image, (b) enhanced using the autoencoder, (c) ridge quality estimated from (b) and ridge dictionary in Fig. 6, (d) cropping overlaid on the input latent image, (e) ridge flow overlaid on (the input latent image (a)) and (f) ridge spacing shown as a heat map

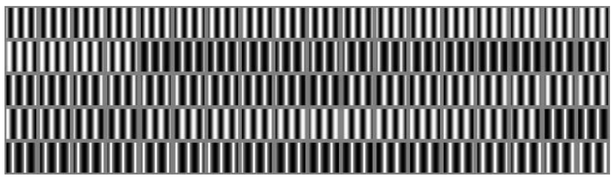


Fig. 6: Ridge structure dictionary (90 elements) for estimating ridge quality, ridge flow and ridge spacing. The patch size of the dictionary elements is 32×32 pixels. Figure retrieved from [30].

trates some of the dictionary elements in vertical orientation with different widths of ridges and valleys.

In order to estimate the ridge flow and ridge spacing, the enhanced latent image output by the autoencoder is divided into 32×32 patches with overlapping size of 16×16 pixels. For each patch P, its similarity s_i with each dictionary element d_i (normalized to mean 0 and s.d. of 1) is computed as $s_i = \frac{P \cdot d_i}{||P||+\alpha}$, where $\cdot$ is the inner product, $|| \cdot ||$ denotes the l_2 norm and α ($\alpha = 300$ in our experiments) is a regularization term. The dictionary element d_m with the maximum similarity s_m ($s_m >= s_i, \forall i \neq m$) is selected and the ridge orientation and spacing of P are regarded as the corresponding values of d_m. The ridge quality of the patch P_I in the input latent image corresponding to P is defined as the sum of s_m and the similarity between P_I and P. Figs. 5 (c), (d) and (f) show the ridge quality, ridge flow and ridge spacing, respectively. Patches with ridge quality larger than s_r ($s_r = 0.35$ in our experiments) are considered as valid fingerprint patches. Morphological operations, including *open* and *close* operations, are used to obtain a smooth cropping. Fig. 5 (d) shows the cropping (ROI) of the latent in Fig. 5 (a).

TABLE 1: The network architecture of autoencoder. Size In and Size Out columns follow the format of $height \times width \times \#channels$. Kernel column follows the format of $height \times width$, $stride$. Conv and Deconv denote convolutional layer and deconvolutional layer (or transposed convolutional layer), respectively.

Layer	Size In	Size Out	Kernel
Input	128×128 × 1	-	-
Conv1	128×128 × 1	64×64 ×16	4×4, 2
Conv2	64×64 ×16	32×32 ×32	4×4, 2
Conv3	32×32 ×32	16×16 ×64	4×4, 2
Conv4	16×16 ×64	8×8 ×128	4×4, 2
Conv5	8×8 ×128	4×4 ×256	4×4, 2
Deconv1	4×4 ×256	8×8 ×128	4×4, 2
Deconv2	8×8 ×128	16×16 ×64	4×4, 2
Deconv3	16×16 ×64	32×32 ×32	4×4, 2
Deconv4	32×32 ×32	64×64 ×16	4×4, 2
Deconv5	64×64 ×16	128×128 ×1	4×4, 2

4 Minutiae Detection via Autoencoder

A convolutional Autoencoder-based minutiae detection approach is proposed in this section. Two minutiae extractor models are trained: one model (*MinuNet_reference*) is trained using manually edited minutiae on reference fingerprints while the other one (*MinuNet_Latent*) is fine-tuned based on *MinuNet_reference* using manually edited minutiae on latent fingerprint images.

4.1 Minutiae Editing

In order to train networks for minutiae extraction for latent and reference fingerprints, a set of *ground truth minutiae* are required. However, marking minutiae on poor quality latent fingerprint images and low quality reference fingerprint images is very challenging. It has been reported that even experienced latent examiners have low repeatability/reproducibility [31] in minutiae markup. To obtain reliable minutiae ground truth, we designed a user interface to show a pair of latent and its corresponding reference fingerprint images side by side; the reference fingerprint image assists in editing minutiae on the latent. The editing tool includes operations of insertion, deletion, and repositioning minutiae points (Fig. 7). Instead of starting markup from scratch, some initial minutiae points and minutiae correspondences were generated using our automated minutiae detector and matcher. Because of this, we refer to this manual process as *minutiae editing* to distinguish it from markup from scratch.

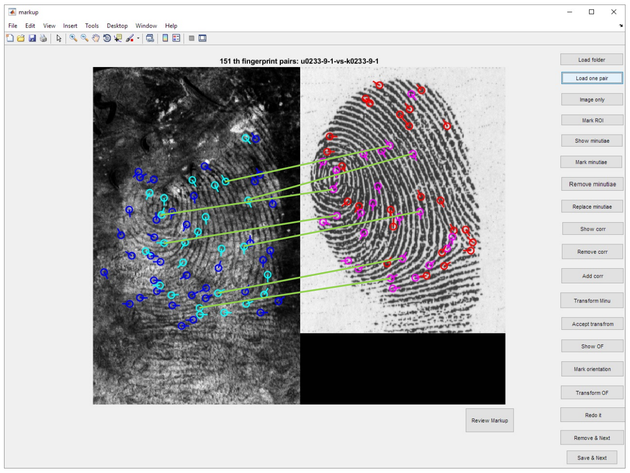


Fig. 7: User interface for minutiae editing. It consists of operators for inserting, deleting, and repositioning minutia points. A latent and its corresponding rolled mate are illustrated here. Only the light blue minutiae in latent correspond with the pink minutiae in the mate shown in green lines.

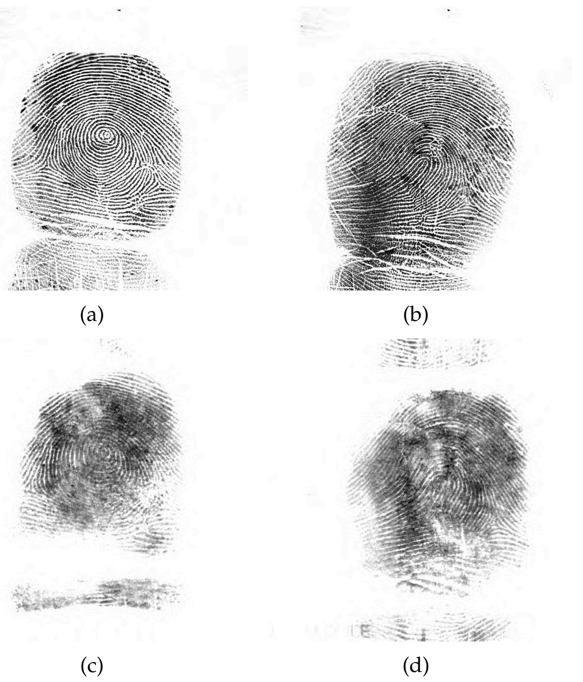


Fig. 8: Examples of rolled fingerprint images from the MSP longitudinal database [32] for training a network for minutiae detection for reference prints. The fingerprint images in the first row are of good quality while the corresponding fingerprint images of the same fingers in the second row are of low quality. The good quality fingerprint images in the first row were used to edit minutiae in poor quality fingerprint images in the second row.

The following *editing protocol* was used on the initially marked minutiae points: i) remove spurious minutiae detected outside the ROI and those erroneously detected due to noise; ii) the locations of remaining minutiae points were adjusted as needed to ensure that they were accurately localized, iii) missing minutiae points which were visible in the image were marked; iv) minutiae correspondences between latent and its rolled mate were edited, including insertion and deletion; A thin plate spline (TPS) model was used to transform minutiae in latent and its rolled mate, and v) a second round of minutiae editing (steps (i)-(iv)) was conducted on latents. One of the authors carried out this editing process.

For training a minutiae detection model for reference fingerprints, i.e., *MinuNet_reference*, a total of 250 high quality and poor quality fingerprint pairs from 250 different fingers from the MSP longitudinal fingerprint database [32] were used. A finger was selected if there is an impression (image) of it with the highest NFIQ 2.0 value Q^h and the lowest NFIQ 2.0 value Q^l which satisfies the following criterion $(Q^h - Q^l) > 70$. This ensured that we can obtain both high quality and low quality images for the same finger (See Fig. 8). A COTS SDK was used to get the initial minutiae and correspondences between selected fingerprint image pairs.

Given the significant differences in the characteristics of latents and rolled reference fingerprints, we fine-tuned the *MinuNet_reference* model using minutiae in latent fingerprint images. A total of 300 latent and reference fingerprint pairs from the MSP latent database were used for retraining. The minutiae detection model *MinuNet_reference* was used to extract initial minutiae points and a graph based minutiae matching algorithm proposed in [25] was used to establish initial minutiae correspondences.

4.2 Training Minutiae Detection Model

Fig. 9 shows a convolutional autoencoder-based network for minutiae detection. The advantages of this model include: i) a large training set since the image patches can be input to the network instead of the whole images , and ii) generalization of the network to fingerprint images larger than the patches. In order to handle the variations in the number of minutiae in fingerprint patches, we encode the minutiae set as a 12-channel minutiae map and pose the training of minutiae detection model as a regression problem.

A minutia point m is typically represented as a triplet $m = (x, y, \theta)$, where x and y specify its location, and θ is its orientation (in the range $[0, 2\pi]$). Inspired by minutia cylinder-code [33], we encode a minutiae set as a c-channel heat map and pose the minutiae extraction as a regression problem (c=12 here). Let h and w be the height and width of the input fingerprint image I and $T = \{m_1, m_2, ..., m_n\}$ be its ISO/IEC 19794-2 minutiae template with n minutiae points, where $m_t = (x_t, y_t, \theta_t)$, $t = 1, ..., n$. Its minutiae map $H \in R^{h \times w \times 12}$ is calculated by accumulating contributions from each minutiae point. Specifically, for each point (i, j, k), a response value $M(i, j, k)$ calculated as

$$M(i, j, k) = \sum_{t=1}^{n} C_s((x_t, y_t), (i, j)) \cdot C_o(\theta_t, 2k\pi/12) \quad (1)$$

where the two terms $C_s((x_t, y_t), (i, j))$ and $C_o(\theta_t, 2k\pi/12)$ are the spatial and orientation contributions of minutia m_t to image point (i, j, k), respectively. $C_s((x_t, y_t), (i, j))$ is defined as a function of the Euclidean distance between

(x_t, y_t) and (i, j):

$$C_s((x_t, y_t), (i, j)) = exp(-\frac{||(x_t, y_t) - (i, j)||_2^2}{2\sigma_s^2}), \quad (2)$$

where σ_s is the parameter controlling the width of the Guassian. $C_o(\theta_t, 2k\pi/12)$ is defined as a function of the difference in orientation value between θ_t and $2k\pi/12$:

$$C_o(\theta_t, 2k\pi/12) = exp(-\frac{d\phi(\theta_t, 2k\pi/12)}{2\sigma_s^2}), \quad (3)$$

and $d\phi(\theta_1, \theta_2)$ is the orientation difference between angles θ_1 and θ_2:

$$d\phi(\theta_1, \theta_2) = \begin{cases} |\theta_1 - \theta_2| & -\pi \leq \theta_1 - \theta_2 < \pi, \\ 2\pi - |\theta_1 - \theta_2| & \text{otherwise.} \end{cases} \quad (4)$$

Fig. 10 illustrates 12-channel minutiae map, where the bright spots indicate the locations of minutiae points. This autoencoder architecture used for minutiae detection is similar to the autoencoder for latent enhancement with parameters specified in Table 1. The three differences are thati i) the input fingerprint patches are size of 64×64 pixels, ii) the output is a 12-channel minutiae map rather than a single channel fingerprint image, and ii) the number of of convolutional layers and deconvolutional layers are 4 instead of 5.

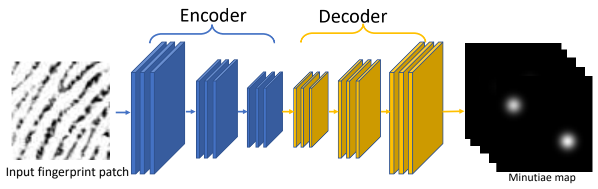


Fig. 9: Training convolutional autoencoder for minutiae extraction. For each input patch, the output is a 12-channel minutia map, where the ith channel represents the minutiae's contributions to orientation $i \cdot \pi/6$.

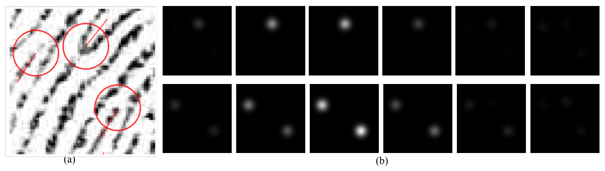


Fig. 10: An example of a minutiae map. (a) Manually marked minutiae overlaid on a fingerprint patch and (b) 12-channel minutiae map. The bright spots in the channel images indicate the location of minutiae points while the channel indicates the minutiae orientation.

The two minutiae detection models introduced earlier, *MinuNet_reference* and *MinuNet_Latent*, are trained. For reference fingerprint images, the unprocessed fingerprint patches are used for training. On the other hand, latent fingerprint images were processed by short-time Fourier transform (STFT) for training in order to alleviate the differences in latents; the model *MinuNet_Latent* is a fine-tuned version of the model *MinuNet_reference*.

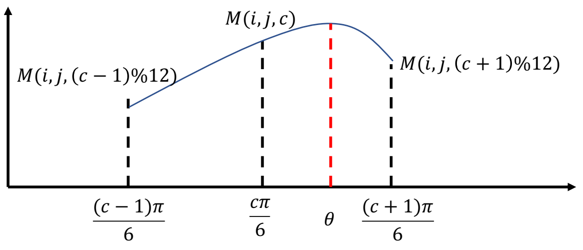


Fig. 11: Minutia orientation (θ) extraction using quadratic interpolation.

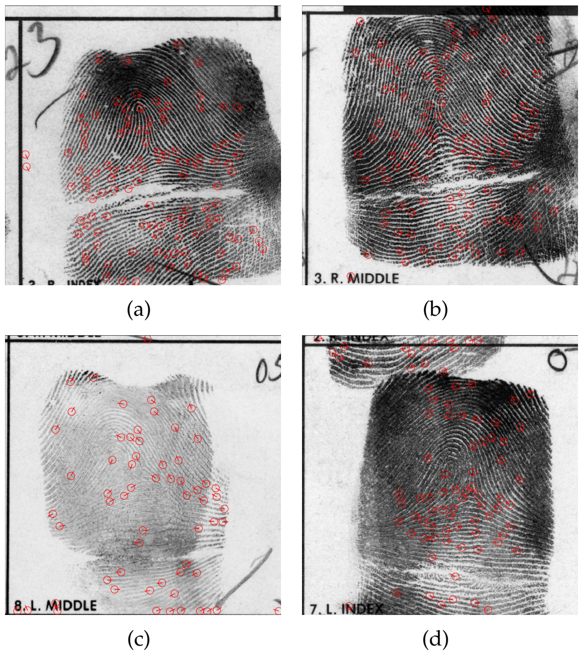


Fig. 12: Examples of minutiae extracted on reference fingerprint images. Images in the first row are of good quality while the images in the second row are of poor quality.

4.3 Minutiae Extraction

Given a fingerprint image of size $w \times h$ in the inference stage, a $w \times h \times 12$ minutiae map M is output by a minutiae detection model. For each location (i, j, c) in M, if $M(i, j, c)$ is larger than a threshold m_t and it is a local max in its neighboring $5 \times 5 \times 3$ cube, a minutia is marked at location (i, j). Minutia orientation θ is computed by maximizing the quadratic interpolation based on $f((c-1) \cdot \pi/6) = M(i, j, (c-1)\%12)$, $f(c \cdot \pi/6) = M(i, j, c)$ and $f((c+1) \cdot \pi/6) = M(i, j, (c+1)\%12)$, where $a\%b$ denotes a modulo b. Fig. 11 illustrates minutia orientation estimation from the minutiae map. Fig. 12 shows some examples of minutiae extracted in reference fingerprints.

5 Minutia Descriptor

A minutia descriptor contains attributes of the minutia based on the image characteristics in its neighborhood. Salient descriptors are needed to establish robust and accurate minutiae correspondences and compute the similarity between a latent and reference prints. Instead of specifying the descriptor in an ad hoc manner, Cao and Jain

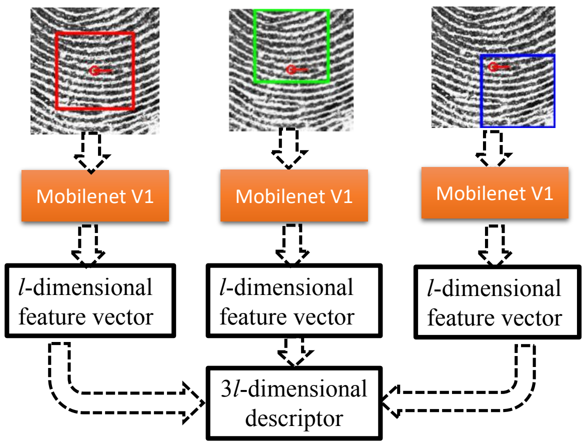

Mobilenet V1
Mobilenet V1
Mobilenet V1
l-dimensional feature vector
l-dimensional feature vector
l-dimensional feature vector
3l-dimensional descriptor

Fig. 13: Extraction of minutia descriptor using CNN.

[25] showed that descriptors learned from local fingerprint patches provide better performance than ad hoc descriptors. Later they improved both the distinctiveness and the efficiency of descriptor extraction [26]. Fig. 13 illustrates the descriptor extraction process. The outputs ($l-$dimensional feature vector) of three patches around each minutia are concatenated to generate the final descriptor with dimensionality $3l$. Three values of l (i.e., l=32, 64, and 128), were investigated; we empirically determine that $l = 64$ provides the best tradeoff between recognition accuracy and computational efficiency. In this paper, we adopt the same descriptor as in [26] , where the descriptor length $L = 192$.

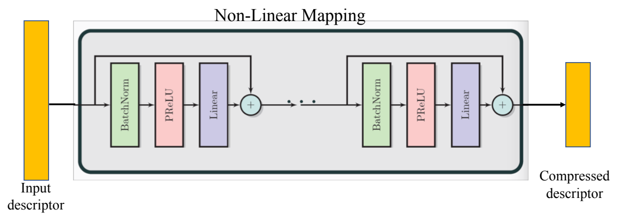


Fig. 14: Framework for descriptor length reduction [27] which reduces descriptor length from 192 to 96.

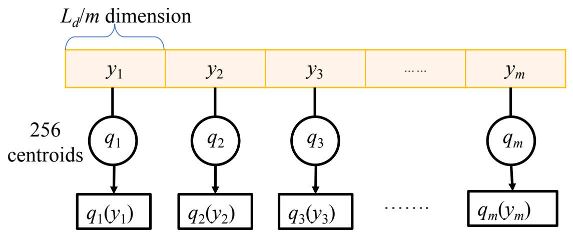
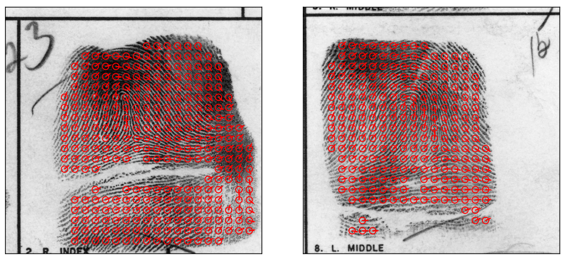


Fig. 15: Illustration of descriptor product quantization.

Since there are a large number of virtual minutiae ($\sim 1,000$) in a texture template, further reduction of descriptor length is essential for improving the comparison speed between input latent and 100K reference prints. We utilized the non-linear mapping network of Gong et al. [27] for dimensionality reduction. The network consists of four linear layers (see Fig. 14), where the objective is to minimize the distance between the cosine similarity of two input descriptors and the corresponding cosine similarity of two output compressed descriptors. Empirical results show that the best value of the descriptor length in the compressed domain (L_d) in terms of recognition accuracy is 96. In order to further reduce the virtual minutiae descriptor length, product quantization is adopted. Given a L_d-dimensional descriptor y, it is divided into m subvectors, i.e., $y = [y_1|y_2|...|y_m]$, where each subvector is of size L_d/m. The quantizer q contains m subquantizers i.e., $q(y) \mapsto [q_1(y_1)|q_2(y_2)|...|q_m(y_m)]$, where each subquantizer quantizes the input subvector into the closest centroid out of the 256 centroids trained by k-means clustering. Fig. 15 illustrates the product quantization process. The distance $D(x, q(y))$ between an input 96-dimensional descriptor x and a quantized descriptor $q(y)$ is computed as



(a) (b)

Fig. 16: Virtual minutiae in two rolled prints; stride size s=32.

$$D(x, q(y)) = \sum_{i=1}^{m} ||x_i - c^i_{q(y_i)}||, \tag{5}$$

where x_i is the ith subvector of x, $c^i_{q(y_i)}$ is the $q(y_i)$th centroid of the ith subvector and $||\cdot||$ is the Euclidean distance. The final dimensionality of the descriptor of rolled prints is $m = 16$.

6 Reference Template Extraction

Given that the quality of reference fingerprints, on average, is significantly better than latents, a smaller number of templates suffice for reference prints compared to latents. Each reference fingerprint template consists of one minutiae template and one texture template. The model *MinuNet_reference* was used for minutiae detection on reference fingerprints. Since the reference fingerprint images were directly used for training, no preprocessing on the reference fingerprint images is needed. Fig. 12 show some examples of minutiae sets extracted on low quality and high quality rolled fingerprint images. For each minutia, the descriptor is extracted following the approach shown in Fig. 13 with descriptor length reduction via nonlinear mapping in Fig. 14.

A texture template for reference prints is introduced in the same manner as for latents. The ROI for reference prints is defined by the magnitude of the gradient and the orientation field with a block size of 16×16 pixels as in [34]. The locations of virtual minutiae are sampled by raster scan with a stride of s and their orientations are the same as the orientations of its nearest block in the orientation field.

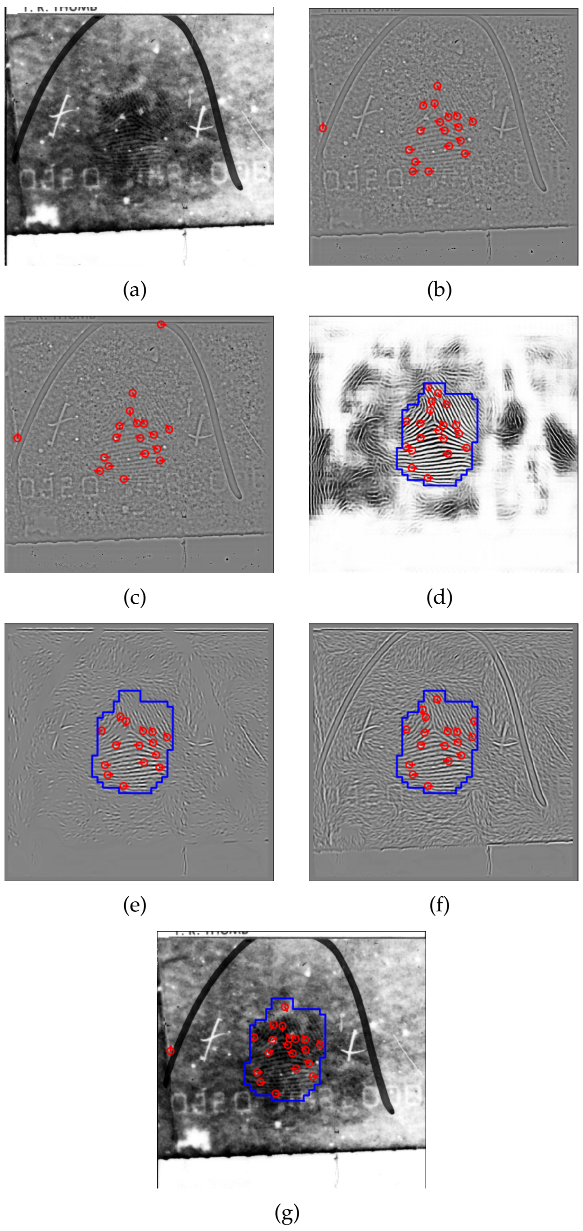


Fig. 17: Latent minutiae extraction. (a) Input latent, (b)-(f) automated extracted minutiae sets after i) STFT based enhancement, ii) autoencoder based enhancement, iii) contrast based enhancement, followed by STFT based enhancement, iv) decomposition followed by Gabor filtering, and v) contrast based enhancement followed by Gabor filtering, respectively, and (g) common minutiae generated from (b)-(e) using majority voting. Minutiae sets in (b), (d) and (g) are selected for matching. Note that minutiae sets in (b) and (c) are extracted by *MinuNet_Latent* but the mask is not used to remove spurious minutiae in case mask is inaccurate.

The virtual minutiae close to the mask border are ignored. Fig. 16 shows virtual minutiae extracted in two rolled prints. Similar to real minutiae, a 96-dimensional descriptor is first obtained using Fig. 13 and Fig. 14, and then further reduced to 16 dimensions using product quantization.

Algorithm 1 Latent template extraction algorithm

```
1: Input: Latent fingerprint image
2: Output: 3 minutiae templates and 1 texture template
3: Enhance latent by autoencoder; estimate ROI, ridge flow
   and ridge spacing
4: Process friction ridges: (i) STFT, (ii) contrast enhancement + STFT, (iii) autoencoder, (iv) decomposition +
   Gabor filtering and (v) contrast enhancement + Gabor
   filtering
5: Apply minutiae model MinuNet_Latent to processed images (i) and (ii) in step 4 to generate minutiae sets 1 (Fig.
   17 (b)) and 2 (Fig. 17 (c))
6: Apply minutiae model MinuNet_reference to processed
   images (iii) - (v) in step 5 to generate minutiae sets 3
   (Fig. 17 (d)), 4 (Fig. 17 (e)) and 5 (Fig. 17 (f))
7: Generate a common minutiae set 6 (Fig. 17 (g)) using
   minutiae sets 1-5
8: Extract descriptors for minutiae sets 1, 3 and 6 to obtain
   the final 3 minutiae templates
9: Generate a texture template using virtual minutiae and
   the associated descriptor
```

7 Latent Template Extraction

In order to extract complementary minutiae sets for latents, we apply two minutiae detection models, i.e., *MinuNet_Latent* and *MinuNet_reference*, to four differently processed latent images as described earlier. This results in five minutiae sets. A common minutiae set (minutiae set 6) is obtained from these five minutiae sets using majority voting. A minutia is regarded as a common minutia if two out of the four minutiae sets contain that minutia, which means the distance between two minutiae locations is less than 8 pixels and the difference in minutia orientation is less than $\pi/6$. Fig. 17 shows these five minutiae sets. For computational efficiency, only minutiae sets 1, 3 and 6 are retained for matching. Each selected minutiae set as well as the set of associated descriptors form a minutiae template. The texture template consists of the virtual minutiae located using ROI and ridge flow [26], and their associated descriptors. **Algorithm** 1 summarizes the latent template extraction process.

8 Latent-to-Reference Print Comparison

Two comparison algorithms, i.e., minutiae template comparison and texture template comparison, are proposed for latent-to-reference comparison (See Fig. 18).

8.1 Minutiae Template Comparison

Each minutiae template contains a set of minutiae points, including their x, y-coordinates and orientations, and their associated descriptors. Let $M^l = \{m_i^l = (x_i^l, y_i^l, \alpha_i^l, d_i^l)\}_{i=1}^{n_l}$ denote a latent minutiae set with n_l minutiae, where (x_i^l, y_i^l), α_i^l and d_i^l are $x-$ and $y-$coordinates, orientation and descriptor vector of the ith minutia, respectively. Let $M^r = \{m_j^r = (x_j^r, y_j^r, \alpha_j^r, d_j^r)\}_{j=1}^{n_r}$ denote a reference print minutiae set with n_r minutiae, where (x_j^r, y_j^r), α_j^r and d_j^r are their $x-$ and $y-$coordinates, orientation and descriptor of the jth reference minutia, respectively. The comparison

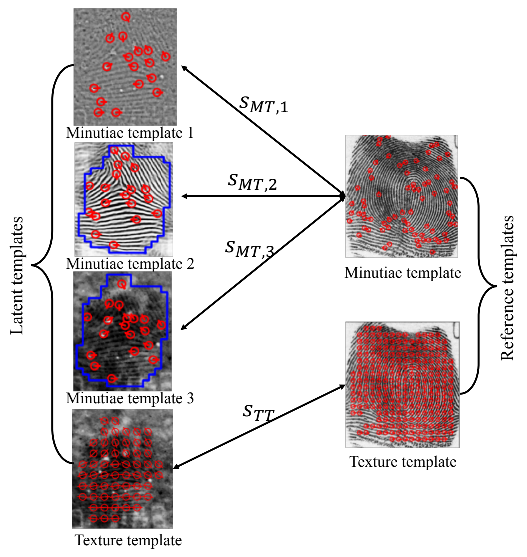


Fig. 18: Latent-to-reference print templates comparison. Three latent minutiae templates are compared to one reference minutiae template, and the latent texture template is compared to the reference texture template. Four comparison scores are fused to generate the final comparison score.

algorithm in [26] is adopted for minutiae template comparison. For completeness, we summarize the minutiae template comparison algorithm in **Algorithm** 2.

Algorithm 2 Minutiae template comparison algorithm

```
1: Input: Latent minutiae template $M^l$ with $n_l$ minutiae and reference minutiae template $M^r$ with $n_r$ minutiae
2: Output: Similarity score
3: Compute the $n_l \times n_r$ similarity matrix ($S$) using the cosine similarity between descriptors
4: Normalize the similarity matrix from $S$ to $S'$ using the approach in [35]
5: Select the top $N$ ($N$=120) minutiae correspondences based on the normalized similarity matrix
6: Remove false minutiae correspondences using simplified second-order graph matching
7: Remove additional false minutiae correspondences using full second-order graph matching
8: Compute similarity $s_{mt}$ between $M^l$ and $M^r$
```

8.2 Texture Template Comparison

Similar to the minutiae template, a texture template contains a set of virtual minutiae points, including their x, y-coordinates and orientations, and associated quantized descriptors. Let $T^l = \{m_i^l = (x_i^l, y_i^l, \alpha_i^l, d_i^l)\}_{i=1}^{n_l}$ and $T^r = \{m_j^r = (x_j^r, y_j^r, \alpha_j^r, d_j^r)\}_{j=1}^{n_r}$ denote a latent texture template and a reference texture template, respectively, where d_i^l is a 96-dimensional descriptor of the ith latent minutia and d_j^r is the $96/m$-dimensional quantized descriptor of the jth reference minutia. The overall texture template comparison algorithm is essentially the same as the minutiae template comparison algorithm in **Algorithm** 2 with two main differences: i) descriptor similarity computation and ii) top N virtual minutiae correspondences selection. The similarity $s(d_i^l, d_j^r)$ between d_i^l and d_j^r is computed as $s(d_i^l, d_j^r) = D_0 - D(d_i^l, d_j^r)$, where D_0 is a threshold and $D(d_i^l, d_j^r)$ is defined in Eq. (5) which can be computed offline.

Instead of normalizing all scores and then selecting the top N ($N = 200$ for texture template comparison) initial virtual minutiae correspondences among all $n_l \times n_r$ possibilities, we select the top 2 reference virtual minutiae for each latent virtual minutiae based on virtual minutiae similarity and select the top N initial virtual minutiae correspondences among $2 \cdot n_l$ possibilities ($2 \cdot n_l$ correspondences are all selected if $2 \cdot n_l <= N$). In this way, we further reduce the computation time.

8.3 Similarity Score Fusion

Let $s_{MT,1}$, $s_{MT,2}$ and $s_{MT,3}$ denote the similarities between the three latent minutiae templates against the single reference minutiae template. Let s_{TT} denote the similarity between the latent and reference texture templates. The final similarity score s between the latent and the reference print is computed as the weighted sum of $s_{MT,1}$, $s_{MT,2}$, $s_{MT,3}$ and s_{TT} as below:

$$s = \lambda_1 s_{MT,1} + \lambda_2 s_{MT,2} + \lambda_3 s_{MT,3} + \lambda_4 s_{TT}, \qquad (6)$$

where λ_1, λ_2, λ_3 and λ_4 are the weights that sum to 1; their values are empirically determined to be 1, 1, 1 and 0.3, respectively.

8.4 Implementation

Both minutiae template comparison and texture template comparison algorithms are implemented in C++. In addition, matrix computation tool Eigen[8] is used for faster minutiae similarity computation. OpenMP (Open Multi-Processing)[9], an application programming interface (API) that supports multi-platform shared memory multiprocessing programming, is used for code parallelization. Hence the latent-to-reference comparison algorithm can be executed on multiple cores simultaneously. The search speed ($\sim$1.0 ms per latent to reference print comparison) on a 24-core machine is able to achieve about 10-times speedup compared to a single-core machine.

9 Experiments

In this report, three latent databases, NIST SD27 [36], MSP and WVU databases are used to evaluate the proposed end-to-end latent AFIS. Table 2 summarizes the three latent databases and Fig. 19 shows some example latents. In addition to the mated reference prints, we use additional reference fingerprints, from NIST SD14 [37] and a forensic agency, to enlarge the reference database to 100,000 for search results reported here. We follow the protocol used in NIST ELFT-EFS [38], [39] to evaluate the search performance of our system.

8. https://github.com/libigl/eigen
9. https://www.openmp.org/resources/openmp-compilers-tools/

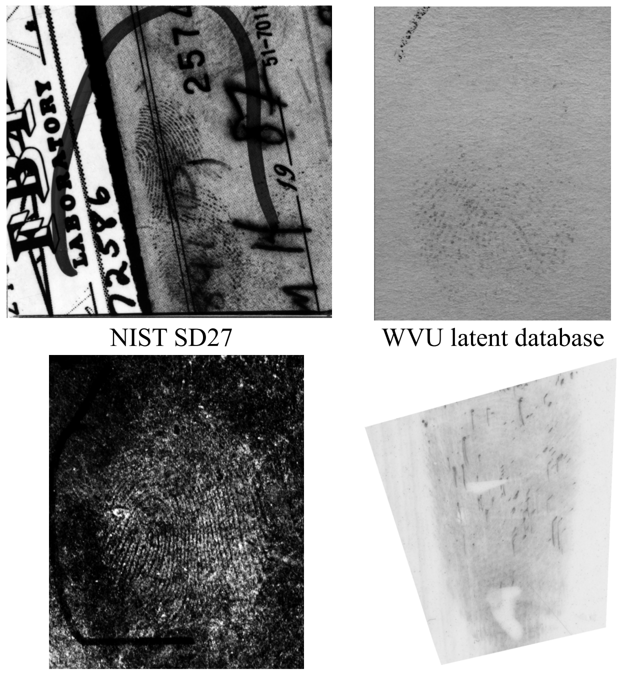


Fig. 19: Examples of latents from the four databases.

TABLE 2: Summary of latent databases.

Database	No. of latents	Source
NIST SD27	258	Forensic agency
MSP	1,200	Forensic agency
WVU	449	Laboratory
N2N	10,000	Laboratory

9.1 Evaluation of Descriptor Dimension Reduction

We evaluate the non-linear mapping based descriptor dimension reduction and product quantization on NIST SD27 against a 10K gallery. Non-linear mapping is adopted to reduce the descriptor length of both real minutiae and virtual minutiae. Three different descriptor lengths, i.e., 128, 96 and 64, are evaluated. Table 3 compares the search performance of different descriptor lengths. There is a slightly drop for 96- and 48-dimensional descriptors, but a significantly drop for 48-dimensional descriptors.

Because of the large number of virtual minutiae, we further reduce the descriptor length of virtual minutiae using product quantization. Table 4 compares the search performance of texture template on NIST SD27 using three different number of subvectors of 96-dimensional descriptors, i.e., $m = 24, 16$ and 12. $m = 16$ achieves a good tradeoff between accuracy and feature length. Hence, we use non-linear mapping to reduce the descriptor length from 192 dimension to 96 dimension and then further reduce virtual minutiae descriptor length to $m = 16$ using product quantization in the following experiments.

TABLE 3: Search performance on NIST SD27 after non-linear mapping

Dimension	Rank-1	Rank-5	Rank-10
192	72.5%	77.5%	79.5%
96	71.3%	77.5%	79.1%
48	61.6%	67.8%	70.9%

TABLE 4: Search performance of texture template on NIST SD27 using different product quantization (PQ) settings.

Value of m	Rank-1	Rank-5	Rank-10
Without PQ	65.5	70.5%	74.8%
m =24	64.3%	69.8%	72.1%
m =16	63.6%	69.4%	71.3%
m =12	58.9%	65.1%	69.8%

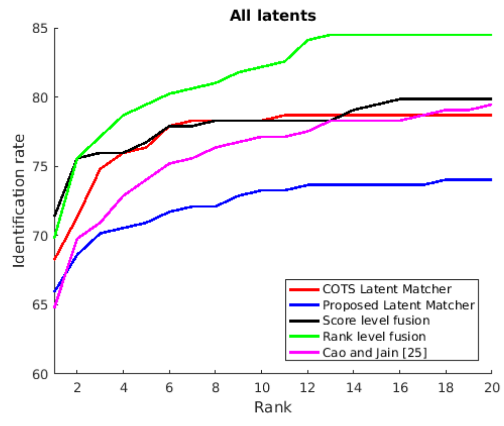
9.2 Search Performance

We benchmark the proposed latent AFIS against one of the best COTS latent AFIS[10] as determined in NIST evaluations. Two fusion strategies, namely score-level fusion (with equal weights) and rank-level fusion (top-200 candidate lists are fused using Borda count), are adopted to determine if the proposed algorithm and COTS latent AFIS have complementary search capabilities. In addition, the algorithm proposed in [25] is also included for comparison on NIST SD27 and WVU databases.

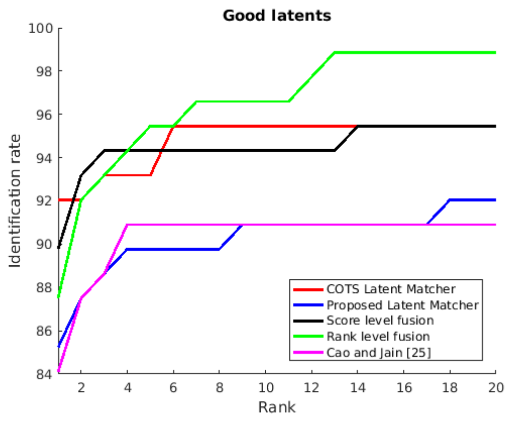
The performance is reported based on close-set identification where the query is assumed to be in the gallery. Cumulative Match Characteristic (CMC) curve is used for performance evaluation. Fig. 20 compares the five CMC curves on all 258 latents in NIST SD27 as well as subsets of latents of three different quality levels (good, bad and ugly) and Fig. 21 compares the four CMC curves on 1,200 latents in MSP latent database. On both operational latent databases, the performance of our proposed latent AFIS is comparable to that of COTS latent AFIS. In addition, both rank-level and score-level fusion of two latent AFIS can significantly boost the performance, which indicates that these two AFIS provide complementary information. Figs. 22 (a) and (b) show two examples that our latent AFIS can retrieve their true mates at rank-1 but the COTS AFIS cannot due to overlap between background characters and friction ridges. Figs. 22 (c) and (d) show two failure cases of the proposed latent AFIS due to the broken ridges. The rank-1 accuracy of proposed latent AFIS on NIST SD27 is slightly higher than the algorithm proposed in [25] even though manually marked ROI was used in [25].

The five CMC curves on 449 latents in WVU database are compared in Fig. 23 and the four CMC curves on 10,000 latents in N2N database are compared in Fig. 24. Both WVU and N2N databases were collected in laboratory. The latents in these two latent databases are dry (ridges are broken), and are significantly from operational latents which were used for fine-tuning minutiae detection model and rolled prints which were used for training Autoencoder for enhancement, the minutiae detection model and enhancement model do

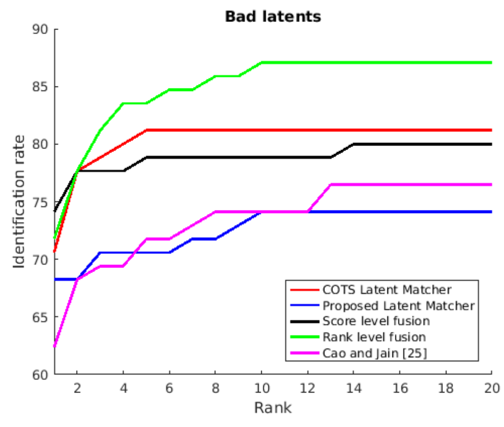
10. The latent COTS used here is one of the top-three performers in the NIST ELFT-EFS evaluations [38], [39] and the method in [25]. Because of our non-disclosure agreement with the vendor, we cannot disclose its name.



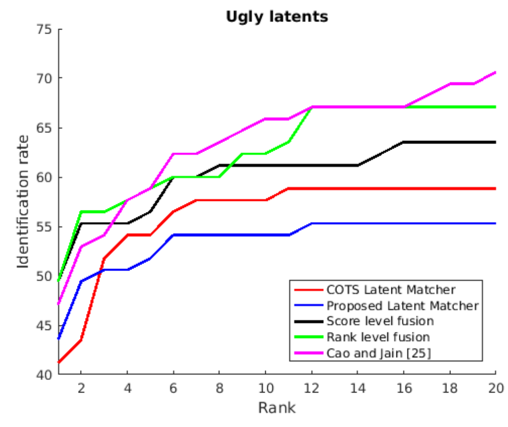
(a)



(b)



(c)



(d)

Fig. 20: Cumulative Match Characteristic (CMC) curves of our latent search system and COTS latent AFIS, their score-level and rank-level fusions, and semi-automatic algorithm of Cao and Jain [25] on (a) all 258 latents in NIST SD27, (b) subset of 88 "good" latents, (c) subset of 85 "bad" latents and (d) subset of 85 "ugly" latents. Note that the scales of the y-axis in these four plots are different to accentuate the differences between the different curves.

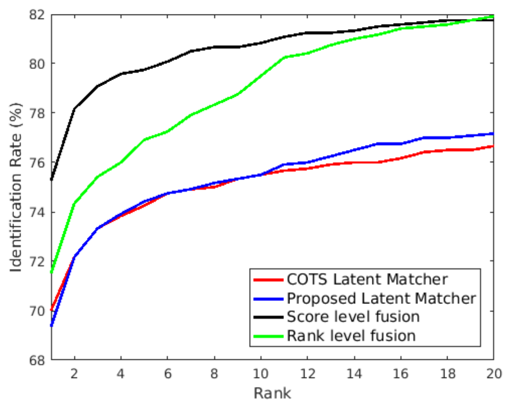


Fig. 21: CMC curves of our latent search system, COTS latent AFIS, and score-level and rank-level fusion of the two systems on the MSP latent database against 100K reference prints.

not work well on WVU latent database. This explains why the performance of the proposed latent AFIS is lower than COTS latent AFIS. Fig. 25 shows some examples where the enhancement model fails. This indicates that additional dry fingerprints are needed for proposed training for deep learning based approaches.

10 Summary

We present the design and prototype of an end-to-end fully automated latent search system and benchmark its performance against a leading COTS latent AFIS. The contributions of this paper are as follows:

- Design and prototype of the first fully automated end-to-end latent search system different curves.
- Autoencoder-based latent enhancement and minutiae detection.
- Efficient latent-to-reference print comparison. One latent search against 100K reference prints can be completed in 100 seconds on a machine with Intel(R) Xeon(R) CPU E5-2680 v3@2.50GHz.

There are still a number of challenges we are trying to address listed below.

- Improvement in automated cropping module. The current cropping algorithm does not perform well on dry latents in WVU and N2N databases.

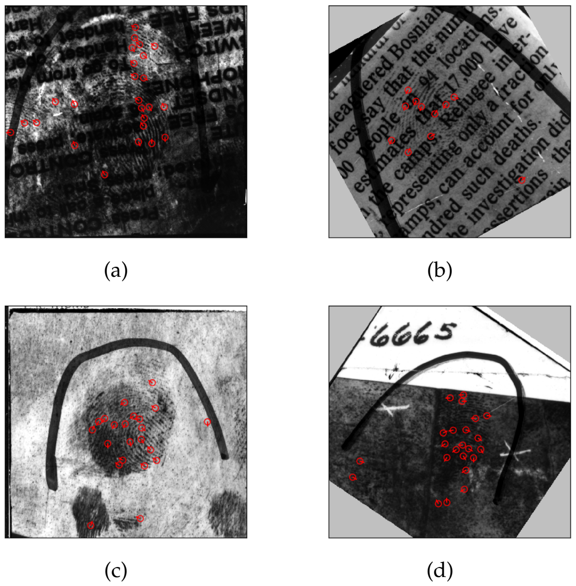


Fig. 22: Our latent AFIS can retrieve the true mates of latents in (a) and (b) at rank-1 which the COTS latent AFIS cannot. COTS latent AFIS can retrieve the mates of latents in (c) and (d) at rank-1 while our latent AFIS cannot. One minutiae set extracted by our AFIS is overlaid on each latent. These latents are from the NIST SD27 database.

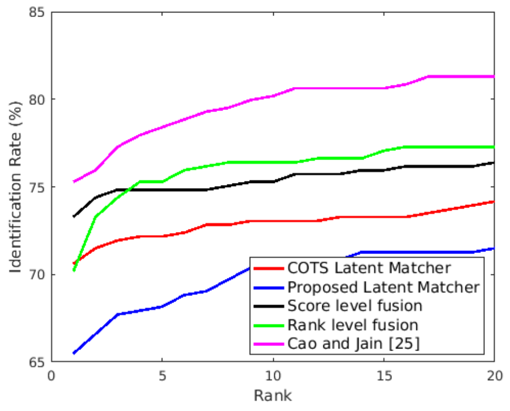


Fig. 23: CMC curves of our latent search system, COTS latent AFIS. Score-level and rank-level fusions of the two systems on the WVU latent database against 100K reference prints show that the bot the fusion schemes boost the overall recognition accuracy significantly.

- Obtain additional operational latent databases for robust training for various modules in the search system.
- Include additional features, e.g., ridge flow and ridge spacing, for similarity measure.

ACKNOWLEDGMENTS

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2018-18012900001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

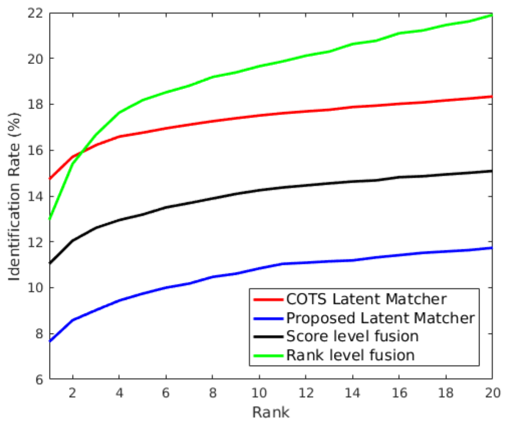


Fig. 24: CMC curves of our latent search system, COTS latent AFIS, and score-level and rank-level fusion of the two systems on the N2N latent database against 100K reference prints.

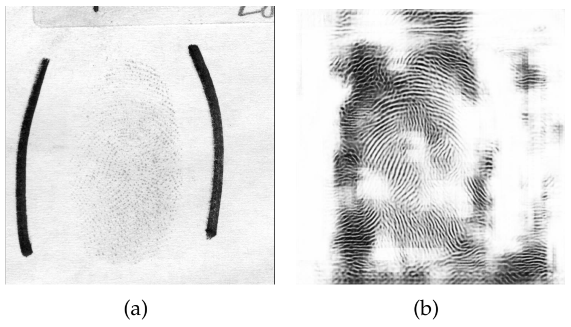


Fig. 25: A failure case in the WVU latent database. Because the training database does not have any dry fingerprints like the latent image in (a), the enhanced latent image in (b) by the Autoencoder does not look good.

REFERENCES

[1] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer, 2009.
[2] "Census of publicly funded forensic crime laboratories," 2014.
[3] "NGI monthly fact sheet," June 2018.
[4] C. Watson, G. Fiumara, E. Tabassi, S. L. Cheng, P. Flanagan, and W. Salamon, "Fingerprint vendor technology evaluation," no. 8034, 2012.
[5] M. Indovina, V. Dvornychenko, R. A. Hicklin, and G. I. Kiebuzinski, "Evaluation of latent fingerprint technologies: Extended feature sets (evaluation #2)," no. 7859, 2012.
[6] President's Council of Advisors on Science and Technology, "Forensic science in criminal courts: Ensuring scientific validity of feature-comparison methods," http://www.crime-scene-investigator.net/forensic-science-in-criminal-courts-ensuring-scientific-validity-of-feature-comparison-methods.html, 2016.
[7] Committee on Identifying the Needs of the Forensic Sciences Community, National Research Council, "Strengthening forensic science in the united states: A path forward," https://www.ncjrs.gov/pdffiles1/nij/grants/228091.pdf, 2009.

[8] A. K. Jain, J. Feng, A. Nagar, and K. Nandakumar, "On matching latent fingerprints," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2008, pp. 1–8.

[9] A. K. Jain and J. Feng, "Latent fingerprint matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 88–100, 2011.

[10] H. Choi, M. Boaventura, I. A. G. Boaventura, and A. K. Jain, "Automatic segmentation of latent fingerprints," in *IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems*, 2012.

[11] J. Zhang, R. Lai, and C.-C. Kuo, "Adaptive directional total-variation model for latent fingerprint segmentation," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1261–1273, 2013.

[12] K. Cao, E. Liu, and A. K. Jain, "Segmentation and enhancement of latent fingerprints: A coarse to fine ridge structure dictionary," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 9, pp. 1847–1859, 2014.

[13] D.-L. Nguyen, K. Cao, and A. K. Jain, "Automatic latent fingerprint segmentation," in *IEEE International Conference on BTAS*, Oct 2018.

[14] K. Cao and A. K. Jain, "Latent orientation field estimation via convolutional neural network," in *International Conference on Biometrics*, 2015, pp. 349–356.

[15] X. Yang, J. Feng, and J. Zhou, "Localized dictionaries based orientation field estimation for latent fingerprints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 955–969, 2014.

[16] J. Feng, J. Zhou, and A. K. Jain, "Orientation field estimation for latent fingerprint enhancement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 54, no. 4, pp. 925–940, 2013.

[17] J. Li, J. Feng, and C.-C. J. Kuo, "Deep convolutional neural network for latent fingerprint enhancement," *Signal Processing: Image Communication*, vol. 60, pp. 52 – 63, 2018.

[18] R. Prabhu, X. Yu, Z. Wang, D. Liu, and A. Jiang, "U-finger: Multi-scale dilated convolutional network for fingerprint image denoising and inpainting," *arXiv*, 2018.

[19] I. Joshi, A. Anand, M. Vatsa, R. Singh, and P. K. S. D. Roy, "Latent fingerprints enhancement using generative adversarial networks," in *to appear in Proceedings of IEEE Winter Conference on Applications of Computer Vision*, 2018.

[20] Y. Tang, F. Gao, and J. Feng, "Latent fingerprint minutia extraction using fully convolutional network," *arXiv*, 2016.

[21] L. N. Darlow and B. Rosman, "Fingerprint minutiae extraction using deep learning," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, Oct 2017, pp. 22–30.

[22] Y. Tang, F. Gao, J. Feng, and Y. Liu, "Fingernet: An unified deep network for fingerprint minutiae extraction," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, Oct 2017, pp. 108–116.

[23] D.-L. Nguyen, K. Cao, and A. K. Jain, "Robust minutiae extractor: Integrating deep networks and fingerprint domain knowledge," in *2018 International Conference on Biometrics (ICB)*, Feb 2018, pp. 9–16.

[24] R. Krish, J. Fierrez, D. Ramos, J. Ortega-Garcia, and J. Bigun, "Pre-registration of latent fingerprints based on orientation field," *IET Biometrics*, vol. 4, pp. 42–52, June 2015.

[25] K. Cao and A. K. Jain, "Automated latent fingerprint recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[26] K. Cao and A. K. Jain, "Latent fingerprint recognition: Role of texture template," in *IEEE International Conference on BTAS*, Oct 2018.

[27] S. Gong, V. N. Boddeti, and A. K. Jain, "On the intrinsic dimensionality of face representation," *arXiv*, 2018.

[28] E. Tabassi, M. A. Olsen, A. Makarov, and C. Busch, "Towards nfiq ii lite: Self-organizing maps for fingerprint image quality assessment," *NISTIR 7973*, 2013.

[29] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints*, Mar. 2016.

[30] K. Cao, T. Chugh, J. Zhou, E. Tabassi, and A. K. Jain, "Automatic latent value determination," in *2016 International Conference on Biometrics (ICB)*, June 2016, pp. 1–8.

[31] B. T. Ulery, R. A. Hicklin, J. Buscaglia, and M. A. Roberts, "Repeatability and reproducibility of decisions by latent fingerprint examiners," *PloS one*, vol. 7, no. 3, p. e32800, 2012.

[32] S. Yoon and A. K. Jain, "Longitudinal study of fingerprint recognition," *Proceedings of the National Academy of Sciences*, vol. 112, no. 28, pp. 8555–8560, 2015.

[33] 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, pp. 2128–2141, 2010.

[34] S. Chikkerur, A. N. Cartwright, and V. Govindaraju, "Fingerprint enhancement using STFT analysis," *Pattern Recognition*, vol. 40, no. 1, pp. 198–211, 2007.

[35] J. Feng, "Combining minutiae descriptors for fingerprint matching," *Pattern Recognition*, vol. 41, no. 1, pp. 342–352, 2008.

[36] "NIST Special Database 27," http://www.nist.gov/srd/nistsd27.cfm.

[37] "NIST Special Database 14," http://www.nist.gov/srd/nistsd14.cfm.

[38] M. D. Indovina, R. A. Hicklin, and G. I. Kiebuzinski, "Evaluation of latent fingerprint technologies: Extended feature sets (evaluation 1)," *Technical Report NISTIR 7775, NIST*, 2011.

[39] M. D. Indovina, V. Dvornychenko, R. A. Hicklin, and G. I. Kiebuzinski, "Evaluation of latent fingerprint technologies: Extended feature sets (evaluation 2)," *Technical Report NISTIR 7859, NIST*, 2012.