 DOI: 10.3217/978-3-85125-524-9-31

Using a U-Shaped Neural Network for minutiae extraction trained from refined, synthetic fingerprints

Thomas Pinetz[1], Daniel Soukup [1], Reinhold Huber-Mörk [1] and Robert Sablatnig[2]

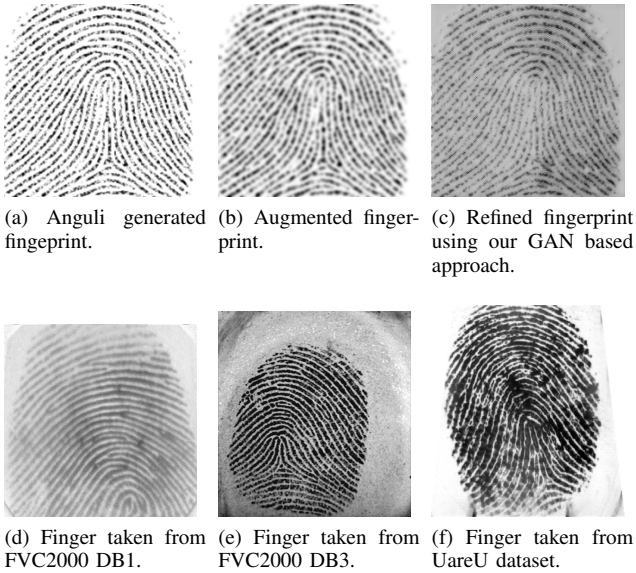
Abstract— **Minutiae extraction is an important step for robust fingerprint identification. However, existing minutia extraction algorithms rely on time consuming and fragile image enhancement steps in order to work robustly. We propose a new approach, combining enhancement and extraction into a Convolutional Neural Network (CNN). This network is trained from scratch using synthetic fingerprints. To bridge the gap between synthetic and real fingerprints, refinements are used. Here, an approach based on Generative Adversarial Networks (GANs) is used to generate fingerprints suited for training such a network and improving its matching score on real fingerprints.**

I. Introduction

Because of their uniqueness and their temporal stability [10], fingerprint minutiae are a reliable way to determine the identity of an individual. Minutiae points are irregularities in ridge patterns, described using coordinates and orientation [17]. Over 150 different irregularities in fingerprints have been identified [18]. While the amount of minutiae on a single fingerprint varies from finger to finger, there are approximately one hundred of such points comprising a regular fingerprint [17]. It was reported that only 10 - 15 minutiae are required to reliably identify an individual [17].

Currently fingerprint matchers like BOZORTH [25] work using minutiae landmarks. Extraction of minutiae is a hard problem though, which heavily relies on good quality fingerprint images [10]. To combat this, image enhancement algorithms are used [4]. Still, reliable minutiae extraction on arbitrary fingerprint images is an open problem as existing feature extractors largely rely on image quality (focus, resolution, skin condition, etc.) [23].

With the rise of deep learning in similar fields [7], [14], [19] and the availability of synthetic fingerprint generators [2], [5], it looks promising to use such methods for minutiae extraction. This paper contributes a new network for minutiae extraction following the idea to solve an equivalent segmentation problem. In this work the synthetic fingerprint generator Anguli [2] is used because of its availability. Anguli generates the training data needed as is shown in Fig. 1a. Because of the difference to real data as visualized in Fig. 1(d-f), augmentations are used (Fig. 1b) as described in Section IV. Here we contribute a novel technique to refine fingerprints based on the GANs [8] paradigm. An example output can be seen in Fig. 1c. Regularization is used to force the refinement network to retain the annotation data while outputting a refined representation of the simulated fingerprint.



(a) Anguli generated fingeprint. (b) Augmented fingerprint. (c) Refined fingerprint using our GAN based approach.

(d) Finger taken from FVC2000 DB1. (e) Finger taken from FVC2000 DB3. (f) Finger taken from UareU dataset.

Fig. 1. Illustration of the fingerprint data used in this work. (a-c are synthetic fingerprints, while d-f are real fingerprints.)

The rest of the paper is organized as follows. Section II reviews related work. In Section III and IV the minutiae extraction algorithm and the refinement method are described in detail. In Section V the results obtained with our method are presented. Finally in Section VI we draw our conclusions.

II. Related Work

Minutiae detection for a sufficiently enhanced image is done by binarization of the grayscale image [10]. Currently fingerprint minutiae extractors use image enhancement routines to achieve the desired quality [10], [4], [25], [24].

Recently there has been a similar approach to the minutiae extraction problem using a pre-trained Convolutional Neural Network, in a forensic setting [23]. However the CNN in [23] is used as a pre-processing step to find large regions containing a minutiae point. Then logistic regression and region pooling are used to extract the actual minutia position.

In our approach the minutia extraction problem is redefined as a binary segmentation task, which the CNN solves directly. With our method there is no need for any time consuming pre- or post-processing. Additionally, synthetic fingerprint generators are used to train the network from scratch and make it suitable for the minutiae extraction problem.

[1]Austrian Institute of Technology, Donau-City-Straße 1, 1220 Wien {thomas.pinetz.fl, daniel.soukup, reinhold.huber-moerk}@ait.ac.at

[2]TU Wien, Karlsplatz 1, 1040 Wien sab@caa.tuwien.ac.at

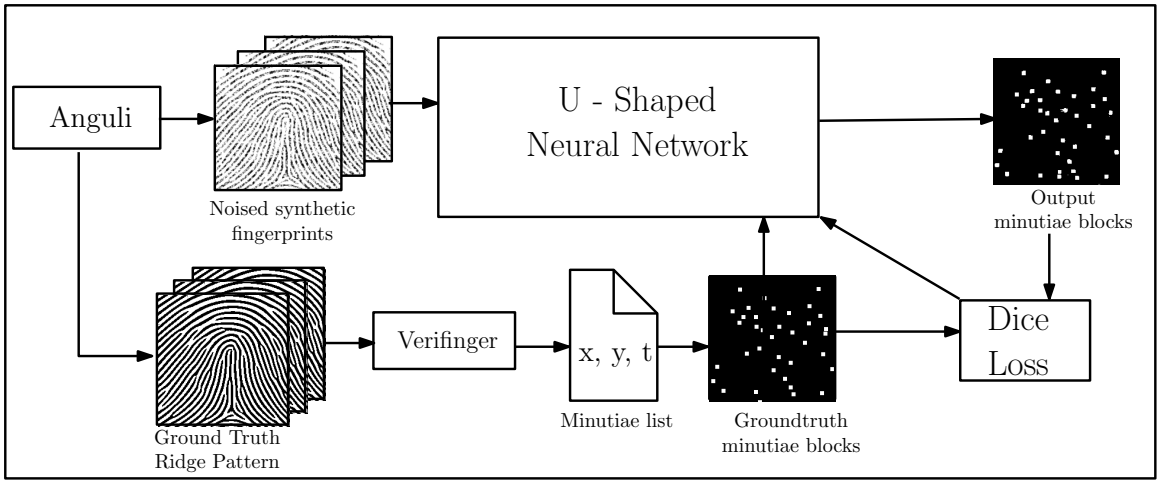


Fig. 2. The whole processing pipeline used for training the minutiae extraction network.

The U-shaped network architecture applied here is used for medical segmentation applications [19], [7]. Training deep neural networks is also the focus of [9], where residual connections are used to allow training of very deep neural networks. Research into making residual connection better is reported in [12], [27], [7].

The GAN framework is first introduced in [8]. Improvements to the stability of adversarial training are proposed in [20], [3]. Based on the results in GANs a refinement network is introduced in [21] for the gaze direction of eye images. In our work a similar approach is used to refine fingerprints.

III. Minutiae Extraction using CNN

The ground truth minutiae list is turned into a binary image by creating an image with the same shape as the corresponding fingerprint and setting every pixel to zero. Then every point in a minutiae region is set to one. A minutia region is defined as a 7×7 pixel square encapsulating the minutia landmark as its centroid. Our deep neural network is used to find a mapping from the input fingerprint to this binary image. This procedure turns the task into a binary segmentation problem.

A. Training Pipeline

The synthetic fingerprint generator Anguli [2] is used to generate a training set. As can be seen in Fig. 2 Verifinger is used to extract the ground truth of the original ridge pattern. For the purpose of this algorithm it is assumed that the minutiae extractor works perfectly on the ridge pattern. Therefore the estimated bifurcations and terminations of the ridge image in Fig. 2 are input to the learning stage as well as ground truth for evaluation. The deviation of the minutiae map and the network output is calculated using dice loss (1), where α is a smoothing factor. Dice loss is reported to produce almost binary outputs [7].

$$loss = -\frac{2y_{pred}y_{true} + \alpha}{\sum y_{pred} + \sum y_{true} + \alpha} \tag{1}$$

B. Network Architecture

The base architecture of the models used in this work can be seen in Fig. 3 and builds on the U-Shaped Network pioneered in [19]. The key differences are:

1) Strided convolution instead of pooling to learn downsampling filters.
2) 224×224 crop to preserve the aspect ratio of the fingerprints.
3) Layer blocks on intermediate levels of the U-Shaped Network instead of pure convolutions.
4) Batch Normalization [11] before every convolution.
5) Dropout with a probability of 0.5 before the final Convolution Layer.
6) Upsampling is done by repeating the pixel in a 2×2 window. Then the upsampling feature maps are concatenated with the output feature maps of the layer block on the same level in the downsampling path. Finally batch normalization, a Regularized Linear Unit (ReLU) activation function and a 3×3 convolution are applied to all the feature maps, before they are passed on to the next layer block.

To preserve information flow, the amount of filters is doubled, when the size of the input data is reduced, as observed in [22]. The layer blocks on specific levels vary in the number of filters used. A model is build with only Wide Residual Blocks [27] (WRN), one with only Densely Connected Blocks [12] (DenseNet) and one with only Bottleneck Residual Blocks [7] (ResUnet). In total, each model used in this work has approximately 8 million parameters.

C. Extracting a Minutiae List

The output of the neural network is a binary minutia regions map. For biometric authentication, a list of minutia points with quality and orientation is needed. For the final position of the minutiae the connected components of the binary map are used. The centroid of each component represents one minutia position. The area a of the connected

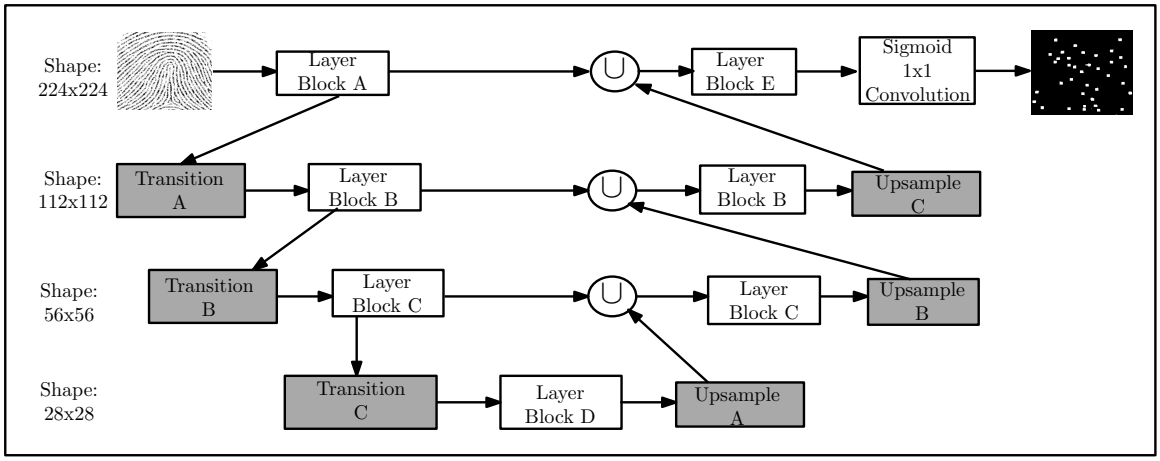
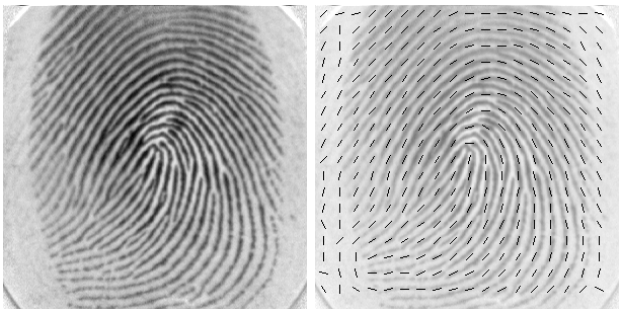


Fig. 3. U-Shaped network architecture used for minutiae extraction.



(a) (b)

Fig. 4. Estimation of the orientation field for a sample fingerprint taken from the FVC2000 DB 1.

(a) (b) (c)

Fig. 5. Anguli [2] generated ridge pattern with two different impressions and the minutiae extracted using Verifinger [24] .

components is used to determine the quality of the minutiae between $0-100$ using $quality = min(a*2,100)$.

The orientation of the minutiae is extracted using an orientation field as described in [10]. The orientation is estimated for every 16×16 region as visualized in Fig. 4b.

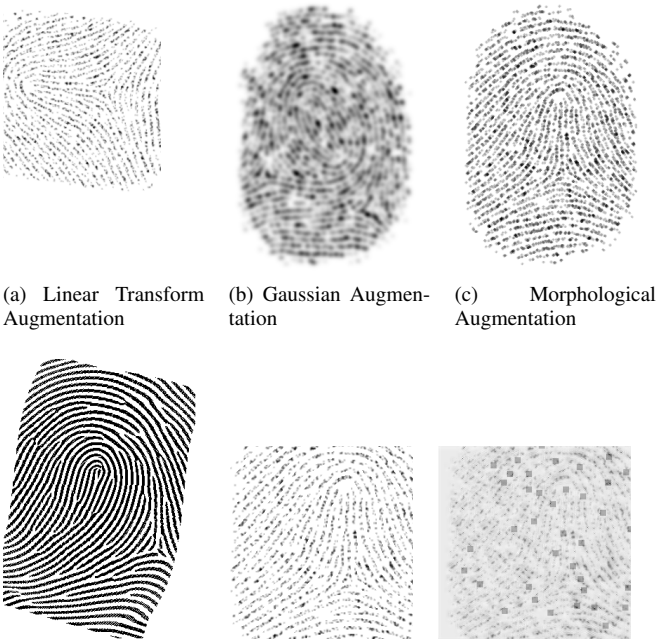
IV. Fingerprint Refinement

The synthetic fingerprint generator Anguli [2] is used to generate random ridge patterns (Fig. 5a). Then multiple variations of every ridge pattern are generated by using different noise models as can be seen in Fig. 5(b,c). Each variation is called an impression of that particular ridge pattern. Because Anguli does not output the minutiae information, a commercial minutiae extractor, Verifinger [24], is used to extract the minutiae data out of the ridge pattern. For the purpose of this paper it is assumed that Verifinger works perfectly on the binary ridge pattern. Those minutiae landmarks are then used for all the impressions (Fig. 5(a-c)).

A. Augmentation on Synthetic Fingerprints

By comparing Fig. 1(d-f) with Fig. 5(a-c) the differences between real and synthetic fingerprints are easily spotted. To bridge this gap the following augmentations are used:

1) **Non linear distortions**: To model the contact region of a fingerprint, random non-linear distortions are used. This also introduces changes in local ridge frequency to synthetic fingerprints as can be seen in Fig. 6d. The distorted ridge pattern is used by Anguli to generate new impressions.
2) **Morphological operations**: Grayscale Dilation and Erosion are used to model wet and dry fingerprint images [5]. An example of this can be seen in Fig. 6c.
3) **Random rotation, translation and shearing**: Fingerprint images are randomly translated, rotated and sheared to gain invariance to linear transformations. An example of this can be seen in Fig. 6a.
4) **Random Blurs**: The images are randomly blurred with a Gaussian kernel, where the variance varies to simulate noisy fingerprints as can be seen in Fig. 6b.
5) **Random Mirroring**: Fingerprint images are randomly mirrored either horizontally or vertically with a 0.5 probability for each direction.
6) **Refinement Network**: A Refinement Neural Network, based on GANs is used to refine images to look more like real world fingerprints. The input size to the network is 224×224. Therefore synthetic fingerprints are resized by a random factor between zero and the difference in image dimension, while keeping the aspect ratio. Then a random 224×224 crop of the



(a) Linear Transform Augmentation (b) Gaussian Augmentation (c) Morphological Augmentation

(d) Distorted Ground Truth Ridge Pattern (e) Refinement Network Input (f) Refinement Network Output with minutiae regions.

Fig. 6. Illustration of the refined data.

resized image is used as input to the network. An example input and output image can be seen in Fig. 6e and Fig. 6f.

B. Refinement Network

The Refinement Network used in this work is based on the GAN paradigm, where a dual optimization problem is solved. A refiner and a discriminator network are simultaneously trained against each other. The refiner network tries to fool the discriminator by applying refinements to a synthetic fingerprint, while the discriminator is used to discriminate between fake and real data. The purpose of such a network is to find a Nash Equilibrium [20] where both networks are optimal.

The only application of a refinement network to our knowledge is in [21]. In our work the approach therein is extended by using noise on the input data to improve the stability of training such a network [3].

One key observation is that using the input image itself for regularization is limiting the amount of possible refinements for fingerprints. Here we propose to use the Hessian of the image instead of the image itself for regularization. The Hessian represents the actual ridge pattern of the fingerprint independent of the pixel intensity values. Mean Squared Error is used to penalize deviation from the Hessian, while the refiner network still needs to fool the discriminator network.

The refiner network uses the same architecture as the minutiae extraction network (Fig. 3), only smaller in the number of layer blocks and filters. Wide residual blocks [27] are used for every layer block starting with 32 filters and doubled on its way down and halved on their way up. Fingerprints like in Fig. 6f are produced by this method. Here, the problem observed by current synthetic fingerprint refiners of modeling noise is addressed by using such a network [5].

V. EXPERIMENTS

This section showcases the results obtained with our method. All our models were programmed using the python framework Keras [6] and trained on a Nvidia Geforce Titan X. For training, the Adam [13] optimizer is used with an initial learning rate of 0.001. The learning rate is cut in half, if the validation error has not decreased for three consecutive epochs. For other minutiae extraction algorithms, an Intel Xeon - W3550 CPU was used.

A. Experimental Setup

For training 28.000 fingerprints with five impressions per fingerprint were generated using Anguli. In total 140.000 fingerprints were used for training, which included a validation set of 10.000 fingerprints. The different impressions can be seen in Fig. 1. Out of the impressions three contain medium noise and the other two use little and heavy noise respectively.

Non linear distortions are used on 3.000 of those fingerprints and on all of their impressions. All the other augmentations, as described in Section IV are applied on the fly.

An annotated real dataset of 300 fingerprints constructed from 220 samples of the sd04 [26] and the 80 images of the fvc2000 DB4_B [15] dataset are used additionally to increase the effectiveness of the classifier. The real dataset used for the refinement network is the UareU [1] dataset.

B. Deep Learning Experiments

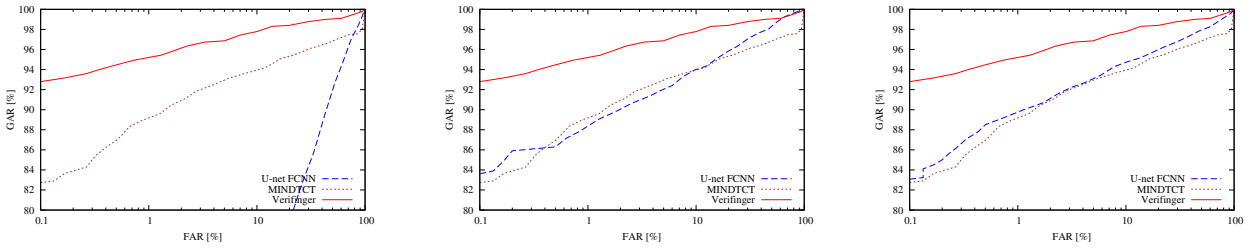
In Fig. 9 the difference in performance for the various layer blocks defined in Section III can be seen. In contrast to the findings in [12] using densely connected blocks did not work as well for the minutiae detection problem. Bottleneck residual blocks performed similarly to wide residual blocks, which is similar to the findings in the original paper [27].

C. Experiments on FVC2000 databases

Here, the performance of our method is compared to other minutiae extraction algorithms on the FVC2000 [15] dataset consisting of real world fingerprints. To match the minutiae against each other, the minutiae matcher BOZORTH [25] was used. The results of this experiment can be seen in Fig. 7 and Fig. 8, where GAR and FAR denote the Genuine Acceptance Rate and False Acceptance Rate accordingly. Using those metrics the Equal Error Rate (EER) can be calculated by finding the rate where (2) holds.

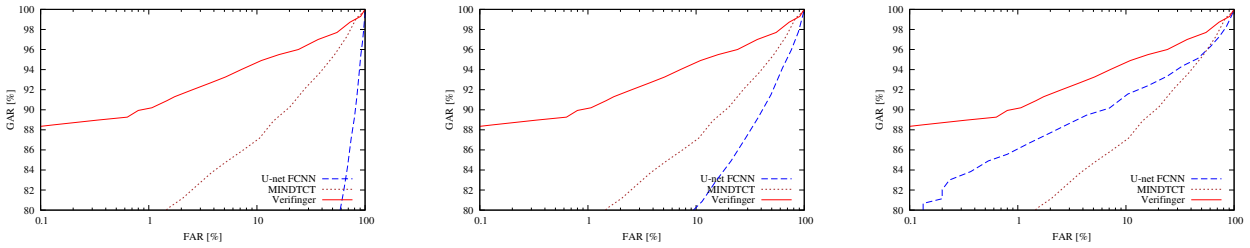
$$GAR = 1 - FAR \tag{2}$$

The extracted EER of the evaluated minutiae extractors is shown in Table I. Our algorithm performs better than



(a) Using only Anguli generated data. (b) Using Augmentations (c) Using Augmentations and Refinements.

Fig. 7. Equal Error Rate Comparison on FVC2000 [15] DB 1 using synthetic, augmentated or refined data.



(a) Using only Anguli generated data. (b) Using Augmentations (c) Using Augmentations and Refinements.

Fig. 8. Equal Error Rate Comparison on FVC2000 [15] DB 3 using synthetic, augmentated or refined data.

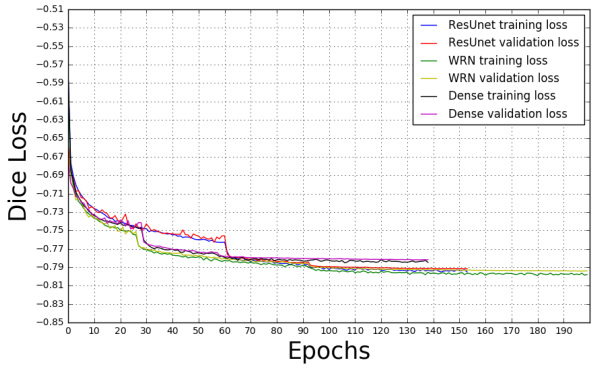


Fig. 9. Model comparision between Dense Blocks, Wide Residual Blocks and Bottleneck Residual Blocks.

MINDTCT on real datasets. Also we report clear performance improvements by using a refinement network. Additionally it is also the fastest method, when run on a GPU.

D. Sample Results for Refinement Network

The only quality metric to our knowledge for GANs is the inception score [20], which is not applicable for our use case. Therefore, this section shows the visual result of the refinement network. In Fig. 10 we can see a comparison of using self regularized MSE versus the Hessian regularized version of the network. In the Hessian regularized examples the ridge pattern is better preserved and less artifacts are introduced into the refined fingerprint.

E. Sample Results for Various Fingers

An illustration on which minutiae are found using different training data is given in Fig. 11. Here, by training solely on

TABLE I

EQUAL ERROR RATE AND ENROLLMENT SPEED FOR FVC2000 [15] DATABASES

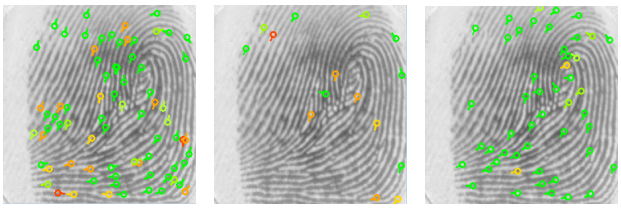
Algorithm	DB 1	DB 3	Time in sec.
Synth. Unet FCNN	21.80%	32.75%	0.12 on gpu
Augm. Unet FCNN	7.01%	16.63%	0.12 on gpu
Ref. Unet FCNN	5.99%	9.42%	0.12 on gpu
MINDTCT [25]	6.63%	12.11%	0.14 on cpu
Verifinger [24]	3.28%	6.31%	1.08 on cpu

(a) Self regularized GAN Output (b) Hessian regularized GAN Output. (c) Hessian regularized GAN Output.

Fig. 10. Sample refiner network output images for self regularized and Hessian regularized training based on the GAN approach.

synthetic fingerprints the minutiae map is clearly wrong as shown in Fig. 11. The network trained on augmented data outputs a subset of the correct minutiae. In contrast, the network trained on GAN data outputs a reasonable minutiae map for this example.

An example of a clear mismatch between two images of the same fingerprint can be seen in Fig. 12. Even though the matching score is 0%, overlapping minutiae are found. However, the orientation does not match because of the noise in the fingerprint.



(a) FCNN trained with synthetic data

(b) FCNN trained with augmented data

(c) FCNN trained on augmented + GAN data.

Fig. 11. Comparison of the output of the same network trained only on synthetic, on augmented and on refined data.

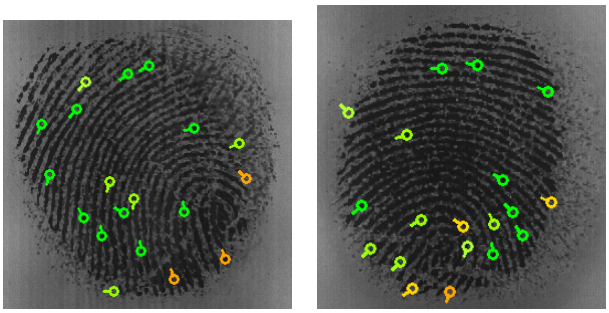


Fig. 12. 0% match of two impressions of the same finger taken from FVC 2002 [16] database with minutiae extracted using the FCNN algorithm.

VI. Conclusion

In this work the possibility of reformulating the fingerprint minutiae extraction problem as a binary segmentation task is shown. Deep learning is used to address this problem. Even with synthetic data as a substitute to annotated real data, the algorithm is able to detect reasonable minutiae with better results than MINDTCT on the FVC2000 dataset without fine tuning of any parameters. Additionally, the performance gain of using our refinement approach was clearly illustrated and advances in training GANs are likely to bring better performance for this minutiae extraction algorithm. A first step is made by using the Hessian instead of the image itself for regularization. However, this performance gain illustrates the dependence on good training data.

Currently, the angle of the minutiae points are calculated using an orientation field. In a future network, we want to learn the orientation of the minutiae by using the orientation field of the ground truth ridge pattern. We believe that better than state-of-the-art performance can be reached using deep learning given sufficiently diverse training data.

ACKNOWLEDGMENT

We thank Peter Wild and Thomas Pock for their helpful insights.

References

[1] "UareU Database," http://www.neurotechnology.com/download.html, 2007, [Online; accessed 01-March-2017].

[2] A. H. Ansari, "Generation and storage of large synthetic fingerprint database," Ph.D. dissertation, Indian Institute of Science Bangalore, 2011.

[3] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, vol. 2016, 2017.

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

[5] 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.

[6] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[7] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer, 2016, pp. 179–187.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[10] L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777–789, 1998.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *arXiv preprint arXiv:1611.09326*, 2016.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[15] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "Fvc2000: Fingerprint verification competition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 402–412, 2002.

[16] ——, "Fvc2002: Second fingerprint verification competition," in *16th international conference on Pattern Recognition. Proceedings.*, vol. 3. IEEE, 2002, pp. 811–814.

[17] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[18] A. A. Moenssens, *Fingerprint techniques*. Chilton Book Company London, 1971.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2226–2234.

[21] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *arXiv preprint arXiv:1612.07828*, 2016.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[23] Y. Tang, F. Gao, and J. Feng, "Latent fingerprint minutia extraction using fully convolutional network," *arXiv preprint arXiv:1609.09850*, 2016.

[24] S. VeriFinger, "Neuro technology (2010)," 2010.

[25] 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 (nbis)," 2007.

[26] C. I. Watson and C. Wilson, "Nist special database 4," *Fingerprint Database, National Institute of Standards and Technology*, vol. 17, p. 77, 1992.

[27] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016. [Online]. Available: http://arxiv.org/abs/1605.07146