# Neural Fingerprint Enhancement

Edward Raff
*Booz Allen Hamilton*
raff_edward@bah.com

***Abstract*—Biometrics fingerprint matching has been done with a heavily hand-tuned and designed process of classical computer vision techniques for several decades. This approach has led to accurate solutions for solving crimes today and, as such, little effort has been devoted to using deep learning in this domain. Given that convolutional neural networks have shown dominance for most other image-based problems, we re-evaluate their potential for improving the fingerprint matching process. By leveraging synthetic data generators, we show that one can train a neural fingerprint enhancer to improve matching accuracy on real fingerprint images. Our approach is both simple in design and for potential deployment and adoption in real world use.**

***Index Terms*—fingerprints, neural networks, biometrics**

## I. Introduction

Deep Learning has become a popular technique for many biometric tasks such as face, iris, handwritten signature, and gait recognition, yet these newer neural network techniques have found relatively little use in fingerprint matching and recognition [29]. Instead, two decades of classical image and signal processing techniques have been applied to the problem with considerable success [23, 24]. By 2006, before any resurgence of neural networks in the machine learning community, these systems where already obtaining low error rates on all but the most challenging examples [3] and became integral to the criminal investigation process.

Fingerprint matching and recognition is generally based around *minutiae* extraction and matching [22]. The minutiae of a fingerprint are the locations where the ridges of a print either ends or combines with another ridge. Minutiae are characterized by their relative location to one another, type, and orientation. These are often represented as a graph, and graph-matching used to compare fingerprint minutiae [11]. At a high level, the standard process for minutiae extraction takes the following steps:

1) Pre-process the image to counter various forms of noise and quality issues.
2) Estimate information about the orientation of the ridges in a fingerprint.
3) Apply a process to identify minutiae locations, and use pre-processed image and orientation data to characterize minutia.
4) Post-process minutiae to remove common spurious, incorrect, and unreliable identifications.

In this work, we seek to re-evaluate the potential for neural networks to add value to the fingerprint matching process by exploiting the deep understanding and modeling that has been done with fingerprint images. In particular, Cappelli [4][2] developed software to generate realistic fingerprint images, allowing the user to set and control different properties of the fingerprint such as rotations, noise, damage, and more. In this work, we will leverage Cappelli's software to generate training data for a denoising convolutional auto-encoder. Specifically, we use a neural network to learn the inverse of the noising process that converts the ground-truth ridge patterns into a noisy and realistic fingerprint. The goal will be to then apply the learned function as a neural fingerprint enhancement (NFE) technique before the minutiae extraction process, similar to how Gabor filters [25] and 2-D Fourier Transforms [5] are often used for fingerprint enhancement today.

We specifically look at enhancing images before processing with existing minutiae extractors and matchers because we believe it is a viable method to achieving real-world adoption. Adoption by law enforcement will be preceded by FBI certification and guidance, court admissibility, extended validation, and a host of other issues. By keeping all of the already accepted technologies in place, and adding one pre-processing step with our NFE, we minimize the risk and effort for law enforcement agencies to test and adopt our technique. It can be used concurrently with any existing product or methods to do continuous testing and, if accepted, can simply sit in-front of already trusted tools. If a failure case occurs, users can simply re-process the fingerprint without our NFE to see if their original tooling would have succeeded.

The rest of our paper is organized as follows. We will address related work in using neural networks in the fingerprint domain in section II. In section III we will review our network design and our methodology to train and evaluate it. Then we will present results showing our NFE improves matching on both unseen real fingerprint images, as well as a challenging synthetic test set, in section IV. Finally we will conclude in section V.

## II. Related Work

The general theme of our work is to use neural networks to model problems that have been understood well enough that computer generated simulations are currently at or near the point of fully specifying the problem space. Shrivastava et al. [28] used Generative Adversarial Networks (GANs) to increase the realism of rendered images of eyes to train a better eye tracking system. By generating the data themselves, they know exact ground truth labels. Tompson et al. [31] used CNNs to reproduce the output of classical 2D and 3D fluid simulations. Their network could produce predictions fast enough for real-time simulations, where the classical methods are too slow for

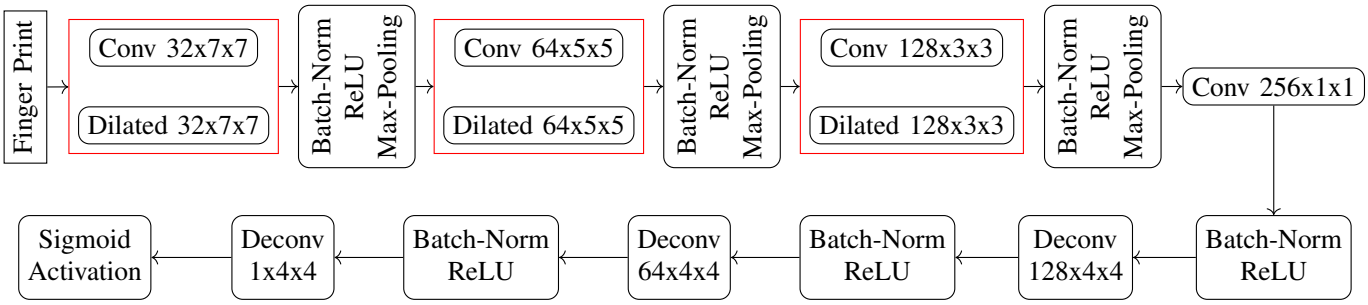



Fig. 1: Diagram of network architecture. Red boxes indicate that two operations are done on the same input, and their results are concatenated together as the final output.

such scenarios. Similar to these prior works, we will be using synthetically generated fingerprints to show the potential for neural networks to learn a more accurate fingerprint image enchantment.

The most similar prior work to our own is by Sahasrabudhe and Namboodiri [26], who trained a convolution Restricted Boltzman Machine (RBM) to try and enhance fingerprint images. They trained in an unsupervised fashion on a hand-picked subset of higher quality fingerprint images from three different databases. Unfortunately, their approach did not generalize to new fingerprints, and most analysis was restricted to the same three datasets that were used for training. We avoid this overfitting problem by using synthetic data and instead train in a supervised fashion to directly learn the denoising goal, and will show that our approach improves performance on new data.

Several works have looked at using multiple neural networks to perform minutiae extraction, combined with additional hand tuned techniques to post-process results by still using classical enhancement techniques like gabor filters [6, 13, 19]. None of these prior works have evaluated the ability to perform accurate matching with these minutia, which is the ultimate end goal. In our work we measure the performance with respect to the goal of accurate matching, and explicitly remove as much domain knowledge and hand-coded image processing from our approach. Work in fingerprint liveness detection (i.e., is the fingerprint from a real finger) has followed a similar path of using considerable hand-designed computer vision pre-processing [20, 33].

Most other works we are aware of that have used neural networks in the fingerprint domain tackle specific sub-goals in larger hard-coded pipelines. Olsen et al. [21] used Self Organizing Maps to evaluate the quality of sub-portions of an image. Yong-xia et al. [36] used neural networks to find the core of a finger print. Sarbadhikari et al. [27] used a simple fully connected network atop FFT based features to classify prints into types (left & right loop, whorl, twin loop, and plain arch).

## III. Methodology

We will now review the methods by which we construct, train, and evaluate our network. The network will be a relatively shallow architecture allowing for fast execution and handling variable input sizes. Training will be done on synthetically generated fingerprints, and evaluation will occur on both real and synthetic prints.

### A. Neural Network Architecture and Training

The overall design of our denoising neural network is given in Figure 1 and is fully convolution. This allows our network to be applied to inputs of any size. This is important since there is no fixed standard size of fingerprint images. Our network starts with of four layers of convolutions, each followed by the use of batch-normalization [9], the ReLU non-linearity [18], and then max-pooling. For each set of convolutions, we split them between the use of standard and dilated convolutions [37]. We start with a total of 64 filters split between each convolution type, doubling the number of filters after each round of pooling. The outputs of these two forms of convolutions are concatenated, and then given to the next layer of the network. We use the mix of dilation and standard in order to capture both local fine-grained details (standard convolutions), as well as farther non-local details via the dilation since our network is (relatively) shallow compared to modern design. After three rounds of convolution and pooling, we apply 1-by-1 convolutions to perform information sharing across the filters. Then we apply deconvolution [38] several times to get to the correct output size, followed by the sigmoid activation to clamp values into the range of $[0, 1]$ that are appropriate for our images.

### B. Training Data

We use Cappelli's [3] Synthetic Fingerprint Generator (SFinGe) version 4.1 to generate our training data. The SFinGe software allows us to specify a number of parameters regarding the generation process, including to save ground-truth images of where fingerprint ridges start and end. An example of a synthetic fingerprint and the ground-truth image is given in Figure 2. The goal is to train a neural network to take the raw input and learn to produce the ridge pattern. We then use the neural network's prediction of the ridge pattern as an enhanced image to use with standard minutiae matching and extraction approaches.

For our training data we collected 600,000 image pairs for training using SFinGe with all combinations of the

(a) Fingerprint generated with "variable" quality setting.

(b) Ground truth ridge-line pattern of generated print.

Fig. 2: Example of synthetic fingerprint from SFinGe. Left image shows the final output (input to the neural network), right shows the ground-truth ridge pattern (target output of the neural network).

following settings: Background $\in \{\text{None}, \text{Optical}, \text{Capacitive}\}$ and distribution as either "Varying quality and perturbations" or "Very Low Quality". Fingerprints were generated against 50,000 simulated fingers for each combination, with 2 impressions for each finger. Across all "Very Low Quality" generated prints we set the horizontal and vertical translation to $\pm 5\%$ and rotation to $\pm 25°$. All other parameters were left with their default options, which includes an image dots per inch (DPI) of 500.

### C. Minutiae Extraction and Matching

We are applying neural networks only to the task of cleaning/enhancing the input image. As such, it is still necessary to use other tools to perform the minutiae extraction and matching process. If our neural fingerprint enhancement is effective, we should see performance improve with the use of any of the tooling. As such, we make use of three tools that are publicly available.

First, we use the NIST MINDTCT program for extraction and BOZORTH3 for matching [34]. We will refer to this pair as *NIST* for short. Despite the age of these tools, a recent study has found them to still be competitive in terms of matching accuracy [22].

Second, we will use the SourceAfis [32] project, which also has a minutiae extraction and matching algorithm. Because SourceAfis is compatible with ISO/IEC 19794-2 [10], we can use the SourceAfis matching algorithm with a different minutiae extractor. DigitalPersona, Inc. released their FingerJetFX commercial solution for minutiae extraction under an open source license [8] . We will denote the use of just SourceAfis for both extraction and matching as *SA*, and the use of FingerJetFX for extraction with SourceAfis for matching as *FJ+SA*.

Both the NIST and FJ extractors uses a 2D block FFT in a similar style as Chikkerur et al. [5]. As such, if we see NFE provide an improvement over FJ+SA and NIST, then we can infer that our approach improves upon the hand turned classical computer vision approach to fingerprint enhancement. Significant prior testing has shown FFT and Gabor based processing to deliver similar high performance [30]. SourceAfis uses its own custom enhancement and ridge smoothing algorithm developed and refined since 2009. Again, our goal is to see that learning the image enhancement process can improve results, thus simplifying the need for fine tuning and maintaining such complex systems for fingerprint matching.

### D. Evaluation on Real Fingerprints

The first set of datasets we will use to validate our approach are real fingerprints from the Fingerprint Verification Competitions (FVC) that were run in 2000 [14], 2002 [15], and 2004 [16]. Each of these competitions has four constituent "databases" (DBs) of prints collected from real people using differing fingerprint readers, and under varying conditions that would introduce noise (such as drying or moistening the fingers) in order to increase the challenge of matching. The fourth database from each of these competitions was synthetically generated using earlier versions of SFinGe. Since our concern is with generalization to real fingerprints, the we exclude the fourth DB from these evaluations. The FVC2004 competition, according to the authors, was meant to be "markedly more difficult than FVC2002 and FVC2000"[1]. A summary of the databases under test is given in Table I.

TABLE I: Fingerprint Verification Competition (FVC) datasets.

| Year | DB# | Sensor Type | Image Size | DPI |
|---|---|---|---|---|
| 2000 | 1 | Low-cost Optical | 300x300 | 500 |
| | 2 | Low-cost Capacitive | 256x364 | 500 |
| | 3 | Optical | 448x478 | 500 |
| 2002 | 1 | Optical | 388x374 | 500 |
| | 2 | Optical | 296x560 | 569 |
| | 3 | Capacitive | 300x300 | 500 |
| 2004 | 1 | Optical | 640x480 | 500 |
| | 2 | Optical | 328x364 | 500 |
| | 3 | Thermal | 300x480 | 512 |

For each database from each competition year, the fingerprints of 10 persons are made publicly available. Each person had their fingerprint taken 8 different times, resulting in a total of 80 images. Due to the small number of total images and persons, we will perform Leave-One-Out (LOO) cross validation on each of the FVC databases from each year. We will record two metrics of interest, the 1-nearest neighbor (1-NN) error (i.e., how often do we fail to return another fingerprint from the same person) and the Area Under the ROC curve (AUC). Because each database has only 10 participants, it is easy for ties to occur. As such the AUC will be our primary focus. The AUC is the sum of area under the trade-off between a false positive rate and a true positive rate, and can be interpreted as the quality of the ranking produced by each method.

Evaluating on real fingerprints from a variety of sensors helps to determine if our method is truly viable and could generalize. Toward this end, we note a number of artifacts that

[1] See http://bias.csr.unibo.it/fvc2004/databases.asp for quote.

exist within the FVC databases that are not modeled by our version of SFinGe. For example, FVC2000 DB1 and DB3 have circular artifacts from the capture device present in each image. FVC2002/DB2 is at a 13% higher DPI than what SFinGe produces and FVC2004/DB3 is using a completely different type of capture device than what SFinGe can currently model. Finally, all images in all databases are at differing image sizes compared to the 256x336 images created by SFinGe.

### *E. Evaluation on Synthetic Fingerprints*

Due to the intrinsic personal nature of fingerprints, there do not currently exist large corpora from which their behavior can be studied. This makes determining smaller impacts on performance difficult. Solving this problem is part of the purpose of SFinGe's development, as it allows us a safe way to generate testing corpora of arbitrary sizes.

As such, we use SFinGe to generate a test corpus with the "Very Low Quality" settings as described in subsection III-B. We generate 100,000 total fingerprints, each with two impressions. The first impression of all 100,000 prints will be enrolled, and querying will be done with the first 1,000 second impressions. This gives us a larger sample size to more accurately determine if NFE improves upon the 1-NN error rate when matching, and if NFE improves matching for the most challenging of fingerprints. We make sure to use a different seed for the test dataset than used in all the training set generations so that the network is not attempting to classifying the "same" finger.

## IV. Results

Training our NFE took only six epochs on a Nvidia Titan X GPU. Only six epochs where used due to a lack of compute resources and time. A batch size of 64 used all 12 GB of RAM, and took 17 hours to train. Reading in a fingerprint, applying NFE, and saving it back to disk as PNG took an average of 0.618 seconds per image. This time could be reduced significantly by applying weight quantization [12] and pruning [17], and integrating matching tools so that the output doesn't need to be written back to a slower HDD[2]. However these improvements are beyond our current scope. Instead we will compare the performance of our three minutiae extraction and matching algorithms on both real and generated test sets.

### *A. Real Fingerprints*

For our evaluation on real fingerprint images, we use the nine databases as discussed in subsection III-D. For each database all three minutiae extraction and matching algorithms, NIST, SA, and FJ+SA, were run with LOO cross-validation twice — first on the original fingerprint images, and then with our Neural Fingerprint Enhancement. Our primary goal is to see that NFE improves the matching accuracy for all of these algorithms as a pre-processing step. All of these results for each algorithm, dataset, and approach can be found int Table II. The third and fourth columns show 1-NN errors and AUC without any additional pre-processing, and the last two columns show the same results but with our NFE pre-processed images.

[2]The HDD overhead causes GPU utilization to average at only 40%

TABLE II: Comparing fingerprint matching on multiple real fingerprint corpora. First column is the year and database number of the FVC competitions. Second column is the base matching algorithm being used. Third and fourth column shows the error rate and AUC when using each algorithm. Last two columns show error and AUC when the image is first pre-processed with our neural finger enhancement. Best results in each row for each metric shown in **bold**, ties show in *italics*.

| | | Standard | | NFE | |
|---|---|---|---|---|---|
| Year/DB# | Algorithm | Errors | AUC | Errors | AUC |
| 2000/1 | NIST | 7 | 92.77 | **1** | **99.00** |
| | SA | 11 | 83.93 | **0** | **99.46** |
| | FJ+SA | 8 | 88.95 | **0** | **99.87** |
| 2000/2 | NIST | 7 | 96.12 | **4** | **97.30** |
| | SA | **3** | 94.48 | 4 | **95.60** |
| | FJ+SA | **0** | **98.23** | 3 | 95.49 |
| 2000/3 | NIST | **2** | 95.05 | 7 | **96.26** |
| | SA | **3** | **88.06** | 8 | 86.35 |
| | FJ+SA | **2** | **93.70** | 10 | 86.81 |
| 2002/1 | NIST | **0** | 97.59 | 3 | **97.97** |
| | SA | *0* | 99.02 | *0* | **99.12** |
| | FJ+SA | *0* | 97.46 | *0* | **99.51** |
| 2002/2 | NIST | 1 | **99.41** | **0** | 99.18 |
| | SA | *0* | **99.88** | *0* | 99.23 |
| | FJ+SA | *0* | **99.84** | *0* | 99.12 |
| 2002/3 | NIST | *1* | 96.56 | *1* | **96.75** |
| | SA | 1 | 97.35 | **0** | **99.23** |
| | FJ+SA | 1 | 96.24 | **0** | **97.92** |
| 2004/1 | NIST | *3* | **95.53** | *3* | 92.88 |
| | SA | *0* | 96.60 | *0* | **96.93** |
| | FJ+SA | *0* | **97.01** | *0* | 96.11 |
| 2004/2 | NIST | 9 | 90.78 | **4** | **92.43** |
| | SA | 3 | 90.85 | **0** | **95.33** |
| | FJ+SA | 7 | 88.83 | **1** | **96.04** |
| 2004/3 | NIST | 4 | 92.43 | **0** | **99.22** |
| | SA | **0** | 95.23 | 2 | **95.28** |
| | FJ+SA | 1 | 96.04 | **0** | **98.83** |

To answer the overall question, "Does NFE improve the AUC of fingerprint matching?", we run a Wilcoxon signed rank test [35]. The Wilcoxon test is preferable to the more common t-test and Friedman test to compare the relative performance of differing algorithms[1, 7]. Running the Wilcoxon signed rank test to compare if the NFE AUC is greater than results without NFE gives us a p-value of 0.020492, a statistically significant result demonstrating that our new approach is an improvement.

Because of the small population size in these test sets, and generally high accuracy of current matching methods, focusing on just the 1-NN results in a number of ties. Some of these ties occur with no 1-NN errors, but still have significantly differing AUC scores. This is because the AUC includes the relative ranking of all seven enrolled fingerprints that belong to the query print. For example, consider a person $\zeta$ with query print $\zeta_q$, and two enrolled prints $\zeta_1$ and $\zeta_2$. If $\zeta_1$ is returned as the

closest fingerprint to $\zeta_q$ and $\zeta_2$ is deemed the farthest (i.e., least similar) fingerprint to $\zeta_q$, this will result in a lower AUC score. Ideally we would see $\zeta_1$ returned as the closest, with $\zeta_2$ returned as the second closest.

Looking at individual tests, we see that NFE provides the largest gains in matching performance on the most difficult corpora. In particular, FVC2000/1 was the most difficult database for all algorithms, with up to 11 errors. NFE reduced the number of errors down to a maximum of 1 error and the AUC improved by up to 15.5 whole points. The test on FVC2004/2 saw similar uniform improvements in performance. Most other datasets had low enough error rates that changes are not necessarily meaningful or resulted in ties, but NFE still tended to improve AUC.

(a) Original print (b) NFE enhanced print

Fig. 3: Example of NFE failure case on the FVC2000/3 database. The database contains a number of properties beyond what SFinGe generates, causing some errors to occur in the enhanced print.

The notable failure of NFE was on the FVC2000/3 database, where NFE had a non-trivial increase in error rates. Manual inspection of these results indicates the failure appears to be caused by the accumulation of features and properties not captured by our version of SFinGe. An example is shown in Figure 3. In the corners of the image we can see that NFE creates some spurious ridges due to the circular receptive field of the fingerprint reader. We can also see spurious valleys be formed within some ridges. This appears to be caused by this dataset containing fingerprint ridges that are generally thicker than what SFinGe produces. Last, we see a false ridge generated at the bottom of the enhanced fingerprint. In the original image there is a gap that appears to be caused by the natural gap between the distal phalanx and middle phalanx. NFE has partially filled in this gap, likely because SFinGe does not generate impressions corresponding to the middle phalanx.

Despite these distributional differences between our generated print and the FVC2000/3 database, it's performance is not completely destroyed and the AUC was even improved for the NIST extractor/matcher. We also believe all of these distributional differences are resolvable. Additional image backgrounds could be created from real fingerprint readers to add to SFinGe's background generation process, the middle phalanx could be simulated using the same kind of process used to generate the distal print, and the maximum pressure/dryness (which affects ridge thickness) could be further increased. In addition, we remind the reader that all of these databases contain properties that differ from SFinGe's generation process. The FVC2004/3 database in particular was collected using an entirely different kind of sensor than what SFinGe models and is at a differing DPI, but NFE still improved the AUC for all three algorithms and reduced the results of two extracttors down to zero errors.

The overall conclusion is that NFE does improve fingerprint matching performance in the majority of cases, and could be further improved by a more effective synthetic distribution generation process. If NFE were deployed, users must ensure that the devices used to capture fingerprints are sufficiently similar to SFinGe's output before use, or improve the generation process to take into account the environmental differences.

### *B. Simulated Fingerprints*

We now look at evaluating NFE's impact on a larger test corpus of low quality fingerprints. As described in subsection III-E we generate a test set of 100,000 "Very low" quality fingerprints to enroll, and a query set of 1,000 fingerprint images. This allows us to better quantify NFE's performance. The results are shown in Table III.

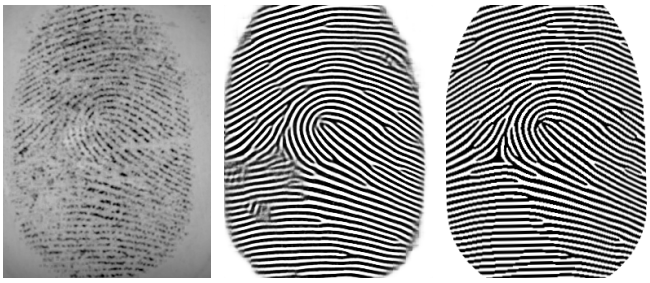
TABLE III: Results on test-set generated with SFinGen "Very low" quality. Second and third column show results without NFE, last two show with NFE. Best results shown in **bold**.

| | Standard | | NFE | |
|---|---|---|---|---|
| Algorithm | Errors | AUC | Errors | AUC |
| NIST | 656 | 91.48 | **255** | **97.81** |
| SA | 721 | 81.91 | **165** | **96.30** |
| FJ+SA | 826 | 78.79 | **173** | **96.05** |

In each case, a marked and dramatic improvement of every metric occurs for every algorithm. The largest impact occurs to the FJ+SA algorithm, reducing 1-NN errors by a factor of 4.8 and increasing the AUC by 17.26 whole points. We note that these improvements in accuracy are on similar scales to those seen on real fingerprints in subsection IV-A on FVC2000/3 and FVC2004/2. This seems to indicate that NFE dramatically improves matching accuracy on the hardest of fingerprints.

An example of one of these low quality fingerprints is given in Figure 4, where we show the print, NFE's enhancement, and the ground truth output. This demonstrates that NFE reduces the complexities and ambiguities that other algorithms and software would need to tackle for the majority of the print. Comparing the result with the ground truth, the majority of minutiae are correctly formed. Most exceptions occur near the edge of the fingerprint, or in the particularly low-quality area near the delta of the fingerprint.

While the AUC indicates uniform improvement of NFE on this corpus, fingerprint matching must often take into account differing false positive rates. As such we show the ROC curve of each method with and without NFE in Figure 5. Here we can see that not only does each algorithm's ROC curve with



(a) Generated print (b) NFE enhanced print (c) SFinGe ground truth

Fig. 4: Example of NFE success on SFinGe "very low" quality test set. Left most image shows the generated fingerprint, middle the result after NFE, and right most the target output.

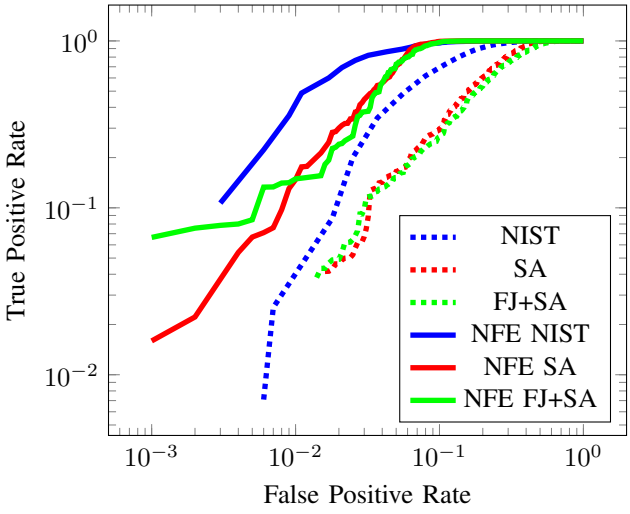



Fig. 5: Receiver operating characteristic curve for all three algorithms with and without NFE on the "very low" quality SFinGe test set. Both axes shown on log scale.

NFE dominate the one without, all of the NFE augmented algorithms dominate *all* non-NFE curves.

## V. Conclusion

By using synthetic training data, we have demonstrated that it is possible to learn a fingerprint enhancement pre-processor without the complex pre-processing steps that have been used in prior works. Our new neural network fingerprint enhancement improves the AUC for multiple different minutiae extraction/matching algorithms on real fingerprints, and testing on additional generated data shows that NFE can significantly improve results on the hardest of fingerprint images. We believe our work shows the potential for further development of neural networks that could simplify fingerprint processing systems today. In future work, we hope to re-visit some of the other tasks which have previously designed neural networks with significant domain knowledge

## References


[1] A. Benavoli, G. Corani, and F. Mangili. Should We Really Use Post-Hoc Tests Based on Mean-Ranks? *Journal of Machine Learning Research*, 17(5):1–10, 2016.

[2] R. Cappelli. SFinGe. In *Encyclopedia of Biometrics*, pages 1–9. Springer US, Boston, MA, 2014.

[3] R. Cappelli, D. Maio, A. Lumini, and D. Maltoni. Fingerprint image reconstruction from standard templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(9):1489–1503, 2007.

[4] R. Cappelli. Synthetic fingerprint generation. *Handbook of Fingerprint Recognition*:203–232, 2003.

[5] S. Chikkerur, A. N. Cartwright, and V. Govindaraju. Fingerprint enhancement using STFT analysis. *Pattern Recognition*, 40(1):198–211, Jan. 2007. ISSN: 0031-3203.

[6] L. N. Darlow and B. Rosman. Fingerprint minutiae extraction using deep learning. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 22–30. IEEE, Oct. 2017. ISBN: 978-1-5386-1124-1.

[7] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, Dec. 2006. ISSN: 1532-4435.

[8] FingerJetFX OSE, 2011. URL: https://github.com/FingerJetFXOSE/FingerJetFXOSE.

[9] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of The 32nd International Conference on Machine Learning*, volume 37, pages 448–456, 2015.

[10] ISO/IEC 19794-2:2011. Technical report, International Organization for Standardization, 2011, page 93.

[11] A. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):302–314, 1997.

[12] D. Lin, S. Talathi, and S. Annapureddy. Fixed Point Quantization of Deep Convolutional Networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2849–2858, New York, New York, USA. PMLR, 2016.

[13] Lu Jiang, Tong Zhao, Chaochao Bai, A. Yong, and Min Wu. A direct fingerprint minutiae extraction approach based on convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 571–578. IEEE, July 2016. ISBN: 978-1-5090-0620-5.

[14] D. Maio, D. Maltoni, R. Cappelli, J. Wayman, and A. Jain. FVC2000: fingerprint verification competition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):402–412, Mar. 2002. ISSN: 01628828.

[15] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. FVC2002: Second Fingerprint Verification

Competition. *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3 - Volume 3*. ICPR '02, 24(3):402–412, 2002.

[16] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. FVC2004: Third fingerprint verification competition. In *Biometric Authentication*, pages 1–7. Springer Berlin Heidelberg, 2004.

[17] D. Molchanov, A. Ashukha, and D. Vetrov. Variational Dropout Sparsifies Deep Neural Networks. In *International Conference on Machine Learning (ICML)*, 2017.

[18] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*:807–814, 2010.

[19] D.-L. Nguyen, K. Cao, and A. K. Jain. Robust Minutiae Extractor: Integrating Deep Networks and Fingerprint Domain Knowledge. In *The 11th International Conference on Biometrics, 2018*, 2018.

[20] R. F. Nogueira, R. D. A. Lotufo, and R. C. Machado. Evaluating software-based fingerprint liveness detection using Convolutional Networks and Local Binary Patterns. In *Proceedings of the IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BIOMS)*, pages 22–29, Rome, Italy, 2014. ISBN: 9781479951765.

[21] M. A. Olsen, E. Tabassi, A. Makarov, and C. Busch. Self-Organizing Maps for Fingerprint Image Quality Assessment. *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*:138–145, 2013. ISSN: 21607508.

[22] D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benítez, H. Bustince, and F. Herrera. A survey on fingerprint minutiae-based local matching for verification and identification: Taxonomy and experimental evaluation. *Information Sciences*, 315:67–87, Sept. 2015. ISSN: 0020-0255.

[23] N. K. Ratha, S. Chen, and A. K. Jain. Adaptive flow orientation-based feature extraction in fingerprint images. *Pattern Recognition*, 28(11):1657–1672, Nov. 1995. ISSN: 00313203.

[24] N. Ratha, K. Karu, Shaoyun Chen, and A. Jain. A real-time matching system for large fingerprint databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):799–813, 1996. ISSN: 01628828.

[25] A. Ross, A. Jain, and J. Reisman. A hybrid fingerprint matcher. *Pattern Recognition*, 36(7):1661–1673, 2003. ISSN: 00313203.

[26] M. Sahasrabudhe and A. M. Namboodiri. Fingerprint Enhancement Using Unsupervised Hierarchical Feature Learning. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, ICVGIP '14, 2:1–2:8, New York, NY, USA. ACM, 2014. ISBN: 978-1-4503-3061-9.

[27] S. N. Sarbadhikari, J. Basak, S. K. Pal, and M. K. Kundu. Noisy fingerprints classification with directional FFT based features using MLP. *Neural Computing & Applications*, 7(2):180–191, 1998. ISSN: 0941-0643.

[28] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251. IEEE, July 2017. ISBN: 978-1-5386-0457-1.

[29] K. Sundararajan and D. L. Woodard. Deep Learning for Biometrics: A Survey. *ACM Comput. Surv.*, 51(3):65:1–65:34, May 2018. ISSN: 0360-0300.

[30] E. Tabassi. Development of NFIQ 2.0. Technical report, National Institute of Standards and Technology, 2012.

[31] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating Eulerian Fluid Simulation With Convolutional Networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3424–3433, International Convention Centre, Sydney, Australia. PMLR, 2017.

[32] R. Važan. SourceAFIS, 2018. URL: https://sourceafis.machinezoo.com/.

[33] C. Wang, K. Li, Z. Wu, and Q. Zhao. A DCNN Based Fingerprint Liveness Detection Algorithm with Voting Strategy. In J. Yang, J. Yang, Z. Sun, S. Shan, W. Zheng, and J. Feng, editors, *Biometric Recognition*, pages 241–249. 2015. ISBN: 978-3-319-25417-3.

[34] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. McCabe, S. Janet, and K. Ki. User's Guide to NIST Biometric Image Software (NBIS). Technical report, National Institute of Standards and Technology, 2007.

[35] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80, Dec. 1945. ISSN: 00994987.

[36] L. Yong-xia, Q. Jin, and X. Rui. A new detection method of singular points of fingerprints based on neural network. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 1, pages 301–305, July 2010.

[37] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations*, 2016.

[38] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.