UNIVERSITY OF CALIFORNIA
RIVERSIDE

Deep Learning for the Analysis of Latent Fingerprint Images

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Jude C. Ezeobiejesi

March 2019

Dissertation Committee:

Dr. Bir Bhanu, Chairperson
Dr. Walid Najjar
Dr. Stefano Lonardi
Dr. Chinya Ravishankar

The Dissertation of Jude C. Ezeobiejesi is approved:

_____

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to express my deep appreciation and thanks to my dissertation advisor, Professor Bir Bhanu, for his guidance and encouragement which made this dissertation possible. I also would like to thank the dissertation committee members, Professors Stefano Lonardi, Walid Najjar, and Chinya Ravishankar, for their feedback and support.

I am forever grateful to my lovely wife Ezinne, my wonderful children Chinenye, Chinazo, and Chike, for their support, patience and understanding throughout the long journey.

Chapter 3 appeared in the 2018 IEEE Computer Vision and Pattern Recognition (CVPR) Workshops, as a paper titled ”Latent fingerprint image quality assessment using deep learning”, co-authored with Bir Bhanu [54].

To my wife and children for all their support and patience.

ABSTRACT OF THE DISSERTATION

Deep Learning for the Analysis of Latent Fingerprint Images

by

Jude C. Ezeobiejesi

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2019
Dr. Bir Bhanu, Chairperson

Latent fingerprints are fingerprint impressions unintentionally left on surfaces at a crime scene. The accuracy of latent fingerprint identification by latent fingerprint forensic examiners has been the subject of increased study, scrutiny, and commentary in the legal system and the forensic science literature. Errors in latent fingerprint matching can be devastating, resulting in missed opportunities to apprehend criminals or wrongful convictions of innocent people. Latent fingerprint comparison is increasingly relied upon by law enforcement to solve crime, and prosecute offenders. The increasing use of this service places new strains on the limited resources of the forensic science delivery system. Currently, latent examiners manually mark the region of interest (ROI) in latent fingerprints and use features manually identified in the ROI to search large databases of reference full fingerprints to identify a small number of potential matches for subsequent manual examination. Given the large size of law enforcement databases containing rolled and plain fingerprints, it is very desirable to perform latent fingerprint processing in a fully automated way.

This dissertation proposes deep learning models and algorithms developed in the context of machine learning for automatic latent fingerprint image quality assessment, quality improvement,

segmentation and matching. Techniques that help speed-up convergence of a deep neural network and achieve a better estimation of the relation between a latent fingerprint image patch and its target class are also proposed. A unified frequency domain based framework for latent fingerprint matching using image patches, as well as a novel latent fingerprint super-resolution model that uses a graph-total variation energy of latent fingerprints as a non-local regularizer for learning optimal weights for high quality image reconstruction, are also presented. Using the deep learning models, this dissertation aims at providing an end-to-end automatic system that solves the problems inherent in latent fingerprint quality assessment, quality improvement, segmentation and matching.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The accuracy of latent fingerprint identification by latent fingerprint forensic examiners has been the subject of increased study, scrutiny, and commentary in the legal system and the forensic science literature. Errors in latent fingerprint matching can be devastating, resulting in missed opportunities to apprehend criminals or wrongful convictions of innocent people. Latent fingerprint image quality assessment provides an indication as to whether the latent fingerprint is a good candidate for further analysis and feature annotations. Figure 1.1 shows latent fingerprints of different qualities. Currently, latent fingerprint examiners assign one of the following values to a given latent fingerprint image: value for individualization (VID), value for exclusion only (VEO), and no value (NV). Latent fingerprints marked as VID have sufficient salient information for matching. Latent fingerprints identified by latent examiners as VEO and NV are generally considered to be valuable and are subject to further processing [29]. As reported by Yoon et. al. [149], 63% of VEO latents in NIST SD27 [7] and WVU [3] latent fingerprint databases can be identified at rank 100 while 40% can be identified at rank 1. Incorrect NV determination for a latent fingerprint could result in missed

opportunity to identify a crime suspect. Latent experts process latent fingerprints by manually marking the regions-of-interest (*ROIs*) in latent fingerprints and using the *ROIs* to search large databases of reference fingerprints and identify a small number of potential matches for manual examination. The poor quality and often complex image background and overlapping patterns characteristic of latent fingerprint images make it very challenging to separate the fingerprint *ROIs* from complex image background and overlapping patterns [153]. In addition, latent fingerprints may contain few minutiae and no singular structures. Matching algorithms that entirely rely on minutiae or alignment of singular structures fail when those structures are missing.

There is also the need to improve the quality of latent fingerprint images to enhance the effectiveness and reliability of matching algorithms. Image image super-resolution (SR) is a technique for generating high-resolution images from low-resolution images by reconstructing the high-frequency components containing details missing from the low-resolution images. Given that there is a natural redundant recurrence of fingerprint patches within the same scale and across different scales of a fingerprint image, it is possible to recover the best possible resolution of each pixel in the image using super-resolution.

This thesis proposes a deep learning model for latent fingerprint quality assessment that eliminates the need for manual feature markup. It presents a unified frequency domain based framework for patch similarity learning, minutiae detection and matching. Similarity scores between patches from the latent and reference fingerprints are determined using a distance metric learned with a convolutional neural network. Minutiae detection in correlated patches is done using a convolutional neural network trained with minutiae patches. The matching score is obtained by fusing the patch and minutiae similarity scores. A novel latent fingerprint super-resolution model

(a) G070L2 **(b)** B122L4 **(c)** U292L1

Figure 1.1: NIST SD27: Latent fingerprints images of different qualities: (a) good, (b) bad, and (c) ugly.

(LAFISR) that uses a graph-total variation energy of latent fingerprints as a non-local regularizer for learning optimal weights for high quality image reconstruction, is also presented. Chapters 2, 3, 4 and 5 present the details of the deep learning models and algorithms developed in the context of machine learning for automatic latent fingerprint image quality assessment, quality improvement, segmentation and matching.

Chapter 2

Data Diffusion for the Segmentation of Latent Fingerprints

Patch based latent fingerprint segmentation falls into the category of tasks where the structure of the input distribution may not reveal enough information about the conditional densities of the target classes from the input examples. For such tasks, deep learning algorithms tend to perform poorly with respect to convergence and minimization of validation errors. Inspired by the principle of information diffusion, a dataset diffusion technique that enables a deep neural network converge faster and achieve a better estimation of the relation between a latent fingerprint image patch and its target class (fingerprint or non-fingerprint) is proposed in this chapter. Experimental results show that using a derived dataset to train and validate a deep neural network for latent fingerprint image segmentation leads to faster convergence of the deep learning algorithm, marked improvement in segmentation accuracy and better generalization of the trained model to unseen examples.

The proposed dataset diffusion technique speeds-up convergence and minimizes valida-

Figure 2.1: Sample latent fingerprints from NIST SD27 showing three different quality levels (a) good, (b) bad, and (c) ugly.

tion errors of a deep neural network for patch based latent fingerprint image segmentation. A similar idea based on the principle of information diffusion has been used by researchers in situations where the neural network failed to converge despite adjustments of weights and thresholds [72], [96]. In [23], the authors described datasets for such tasks as "uncooperative input distributions" and proposed training deep network with "a mixed training criterion that combines the unsupervised objective and a supervised objective".

The rest of the chapter is organized as follows: Section 2.1 presents a review of recent works in data augmentation and latent fingerprint segmentation while section 2.1.2 describes the contributions of this chapter. Section 2.2 highlights our technical approach and presents dataset diffusion as well as an overview of Restricted Boltzmann Machine (RBM), the building block of our deep learning model. Features for complex texture characterization that were used for dataset diffusion are also presented in this section. The experimental results and performance evaluation of our proposed approach are presented in Section 2.3. This section also discusses dataset derivation based on diffusion and highlights the impacts of diffusing the training dataset with the selected texture features on the performance of the network. Section 2.4 contains the conclusions and future work.

2.1 Related Work and Contributions

2.1.1 Related Work

Data augmentation

Data augmentation plays an important role in boosting the performance of object recognition systems. It involves the application of domain-specific transformations to expand a dataset. Data augmentation can be done by using transformations in the data-space to generate additional data samples or creating additional data samples in the feature-space through synthetic over-sampling [143].

Elastic distortion is a data augmentation technique that gives large degrees of freedom in stroke forms, without varying the topological structure of the data [120]. Previous works have used data augmentation techniques based on elastic distortions and affine transformations to improve classifier performance. In [125], Simard et. al applied elastic distortions to existing examples in MNIST handwritten digit dataset to expand the dataset. Using the expanded dataset and convolutional neural networks, they achieved a (2003) state-of-the-art performance (0.4% error) on the MNIST handwritten digit dataset. In [44], the authors performed data augmentation by randomly generating transformations from a set of possible transformations. They demonstrated that data augmentation by elastic distortion gives great boost to classification performance. Xu et. al. [144] improved Relation Classification by Deep Recurrent Neural Networks with data augmentation based on leveraging the directionality of relations. In their work, they used directionality of relationships to create mappings of subject-predicate and object-predicate components of a relation. They used the new data samples that resulted from the mappings to augment the original data samples.

Our work is inspired by the data augmentation techniques used in the works referenced above. Unlike data augmentation which increases the number of data samples, our dataset diffusion approach extends the dimension of the dataset with texture features computed from the dataset.

Segmentation of latent fingerprints

A number of recent studies have been carried out on latent fingerprint segmentation. In [153], [39], [78], and [124], the authors performed latent fingerprint image segmentation by analyzing ridge frequency and orientation properties of the ridge valley patterns to determine the area within a latent fingerprint image that contains the fingerprint. Choi et al. [39], used orientation tensor approach to extract the symmetric patterns of a fingerprint and removed the structural noise in background. They used a local Fourier analysis method to estimate the local frequency in the latent fingerprint image and located fingerprint regions by considering valid frequency ranges. They obtained candidate fingerprint (foreground) regions for each feature (orientation and frequency) and then localized the latent fingerprint regions using the intersection of those candidate regions. Karimi et al. [78] estimated local frequency of the ridge/valley pattern based on ridge projection with varying orientations. They used the variance of frequency and amplitude of ridge signal as features for the segmentation algorithm. They reported segmentation results for only two latent fingerprint images and provided no performance evaluation. Short et al. [124] proposed the ridge template correlation method for latent fingerprint segmentation. They generated an ideal ridge template and computed cross-correlation value to define the local fingerprint quality. They manually selected 6 different threshold values to assign a quality value to each fingerprint block. They neither provided the size and number for the ideal ridge template nor reported an evaluation criteria for the segmentation results. Zhang et al. [153] proposed an adaptive total variation (TV) model for latent

fingerprint segmentation. They adaptively determined the weight assigned to the fidelity term in the model based on the background noise level. They used it to remove the background noise in latent fingerprint images. Cao et al. [31] used ridge structure dictionary to segment latent fingerprint images. Ruangsakul et al. [111] used an algorithm based on spatial-frequency domain analysis to group blocks of latent fingerprints into sub-bands. They sort the subbands and group related spectra to obtain segmentation results. Zhu eta al. [155] used convolutional neural networks to classify multi-sized overlapping patches of a latent fingerprint image as either fingerprint or background. They computed score maps based on the classification results and generated segmentation masks by thresholding the score map to segment the latent fingerprint image.

Our approach uses a deep architecture that learns features from a diffused dataset of latent fingerprint image patches and uses the learnt features to classify the patches into fingerprint and non-fingerprint classes. The patches classified as fingerprint patches are assembled to form the segmented latent fingerprint.

The work described in this chapter has evolved from our earlier preliminary work [53, 55]. In [53], patch based latent fingerprint segmentation was performed by using fractal dimension features computed from latent fingerprint patches to train a weighted extreme learning machine ensemble classifier. The patches were classified into fingerprint and non-fingerprint classes and the final segmentation result was obtained by quilting the patches classified as fingerprint. In [55], we proposed a deep learning model for latent fingerprint segmentation that learns representations of raw latent fingerprint image patches via identity mapping. Features were extracted from the learned representations and used to classify the image patches and the results of the classification were used for latent fingerprint image segmentation. This chapter is different from [53], [55] in the following

aspects: (a) In-depth theoretical and empirical discussions on the selection of features for dataset diffusion are presented. (b) An algorithm for selecting minimal (optimal) number of features used for dataset diffusion to balance model complexity and error on the training data is also presented. (c) Post processing of segmentation results is done via connected component size filtering to obtain segmented regions-of-interest (ROIs) results. (d) Visual segmentation results are presented and compared with the state-of-the-art segmentation results and better quantitative and visual results are obtained. (e) The quality of the segmentation results is evaluated with respect to matching performance, and detailed experimental results are presented. (f) The matching performance of the proposed segmentation method is compared with the state-of-the-art matching results and better results are obtained.

2.1.2 Contributions

The main contributions of this chapter are as follows:

- We proposed a method for diffusing the latent fingerprint dataset using features that characterize complex texture in latent fingerprint images.
- We performed experiments which support the hypothesis that using derived dataset to train and validate the deep network leads to faster convergence of the deep learning algorithm and yields marked improvement in segmentation accuracy compared to current algorithms, as well as better generalization to unseen examples.
- Texture features and variations of texture features have been used for latent fingerprint segmentation. Our approach is the first to use them to extend patch dataset to speed-up convergence and minimize validation errors of a deep neural network for patch based segmentation

of latent fingerprint images.

- We performed detailed experiments for evaluation of segmentation results with respect to the matching performance. These results show better than state-of-the-art performance.

2.2 TECHNICAL APPROACH

Our approach involves partitioning a latent fingerprint image into 8x8 non overlapping blocks and computing texture features for each block. We select optimal subset of texture using deep learning. Using dataset diffusion, we extend the vector of 64 elements representing each block with a vector of the selected features to form a *diffused* dataset. After normalizing the diffused dataset to zero-mean and unit-standard deviation, we learn a set of stochastic features that model a distribution over image patches using a generative multi-layer feature extractor. We use the learned features to train a single hidden layer perceptron classifier that classifies the patches into fingerprint and non-fingerprint classes.

The block diagram of our proposed approach is shown in Figure 2.2. In the patch extraction stage, we extract 8x8 image patches from latent fingerprint images to create a patch dataset. Texture features are computed from the dataset in the compute features stage. In the dataset diffusion stage, the patch dataset is diffused with the features computed in the previous stage. After normalization, the diffused patch dataset is fed to the deep learning model. We used a 7-layer architecture featuring an input layer, 5 hidden layers consisting of Restricted Boltzmann Machines (RBMs) and a single hidden layer perceptron for deep learning, feature extraction and classification, and a two-neuron output layer. In the last stage, we separately assemble the patches classified as fingerprints and non-fingerprint to obtain the segmented fingerprint, and non-fingerprint images,

respectively.

• **Post-Processing:** After quilting the patches classified as fingerprint, we compute the connected components in the resulting image and filter out regions whose area and eccentricity denoted by a_c and e_c, respectively, are outside empirically defined thresholds. The eccentricity of a connected component is defined as the ratio of the distance between the foci of the connected component and its major axis length [9]. It is calculated as

$$e_c = \frac{d_f}{d_v} \tag{2.1}$$

where d_f is the distance from the center to the focus of the connected component, and d_v is the distance from the center to a vertex. We use empirically determined thresholds of $a_c > 100$ and $e_c < 0.5$. A connected component is retained if its $a_c > 100$ and its $e_c < 0.5$. The final segmented fingerprint is obtained after the filtering operation.

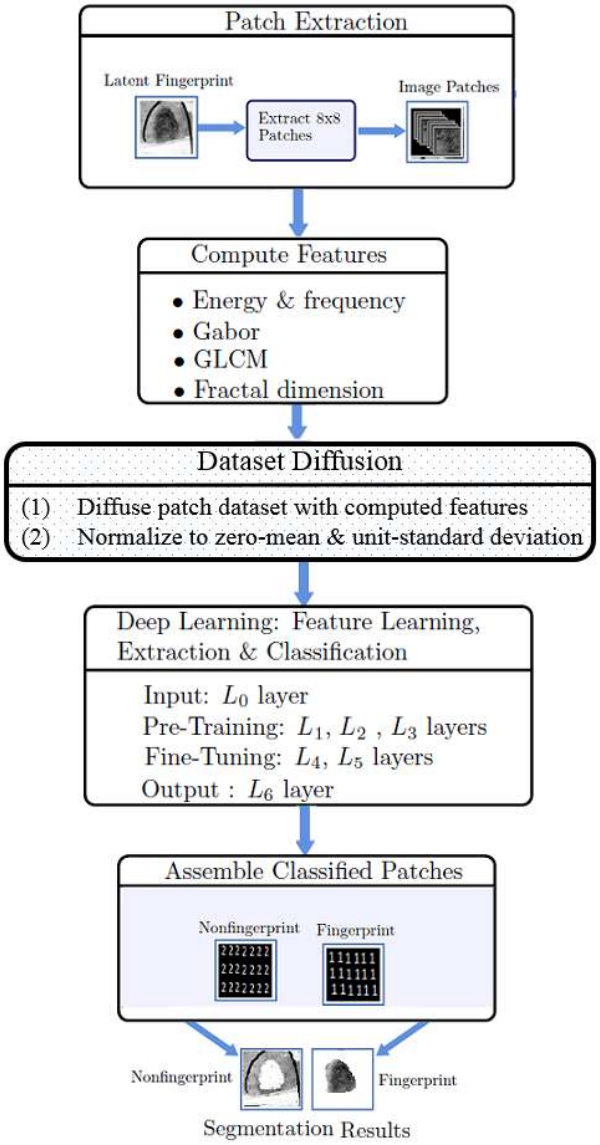


Figure 2.2: Proposed architecture for deep learning, feature extraction and classification. It consists of a stack of RBMs and a single layer perceptron (L_i refers to Layer *i*, $i = 0, \ldots, 6$). It has a receptive field of 9x9, feature dimension of 1,200 and was trained for 50 epochs using a batch size of 100, learning rate of 0.001, and 0.7 momentum. The hyper-parameters and values used to train and validate the model were selected based on the performance of the network on the validation set. The choice of 8x8 patch size is based on its optimality [53].

2.2.1 Dataset Diffusion

Given a dataset $X = \{p_1, p_2, \cdots, p_k\}$, we define the diffusion of *X* as $\hat{X} = \{p_1, p_2, \cdots, x_m\}$, where $m > k$ and each $x_i, k < i < m$ is an element from *n*-tuples of real numbers $(r_1, r_2, \cdots, r_n)$, with the totality of *n*-space denoted as $\mathbb{R}^n$. In other words, $\hat{X}$ is obtained by *extending X* with new elements from $\mathbb{R}^n$. The purpose of dataset diffusion is to minimize reconstruction error and avoid over-fitting during the learning phase of the deep neural network. This leads to a trained model that generalizes well to unseen examples. We perform data diffusion by computing texture features from the patch dataset and selecting the subset of features which when used to diffuse the dataset, yielded the minimum reconstruction error and the best segmentation performance compared to the other candidate feature subsets.

Selecting features for dataset diffusion

We selected features used for dataset diffusion based on the level of their positive contributions to the neuronal activation potential of the neurons in the DANN network. The neuronal activation potential contributions of the features were determined by analyzing the activation function values of the first layer of the our DANN network. The goal was to identify the contribution of each feature to the activation of the neurons that participate in the reconstruction of the input examples. This was done by examining the first hidden layer average activation potential of each feature (over all training examples). Following the analysis, we were able to identify the features that participated positively in the activation of the neurons in the DANN network. The higher the activation potential of a feature, the more likely it will contribute to the learning of input representation, reconstruction and discrimination. We performed 25 training epochs using training and

validation dataset diffused with 20 texture features. Out of 5 models resulting from the training and validation, we selected the model with the minimum input reconstruction and classification errors, and performed activation potential analysis on that model to identify the features that made the most positive contribution to the neuronal activation potentials.

- **Analyzing activation potentials**

The activation potential of k^{th} hidden neuron in a neural network is given by

$$a_k = w_k^T x + b_k \tag{2.2}$$

where w_k is the weight of the out-going connection from the k^{th} neuron, b_k is the bias associated with the k^{th} neuron, and x is the input vector. The activation potential contributed by the j^{th} dimension of N training examples connected to the k^{th} hidden neuron of the first hidden layer of the DANN is given by

$$c_{jk} = \frac{a_k}{N} \tag{2.3}$$

The total contribution of the k^{th} input dimension to the activation potential of the all hidden neurons in the first hidden layer (L_1) of the DANN is given by

$$c_k = \sum_{j=1}^{|L_1|} c_{jk} \tag{2.4}$$

We analyzed the activation potentials of the neurons in the first hidden layer of our network to identify the features to use in diffusing our dataset. The analysis identified the following for each of the 20 features shown in Table 2.1.

- The number of neurons where the feature's contribution to the neurons activation potential is greater than zero.

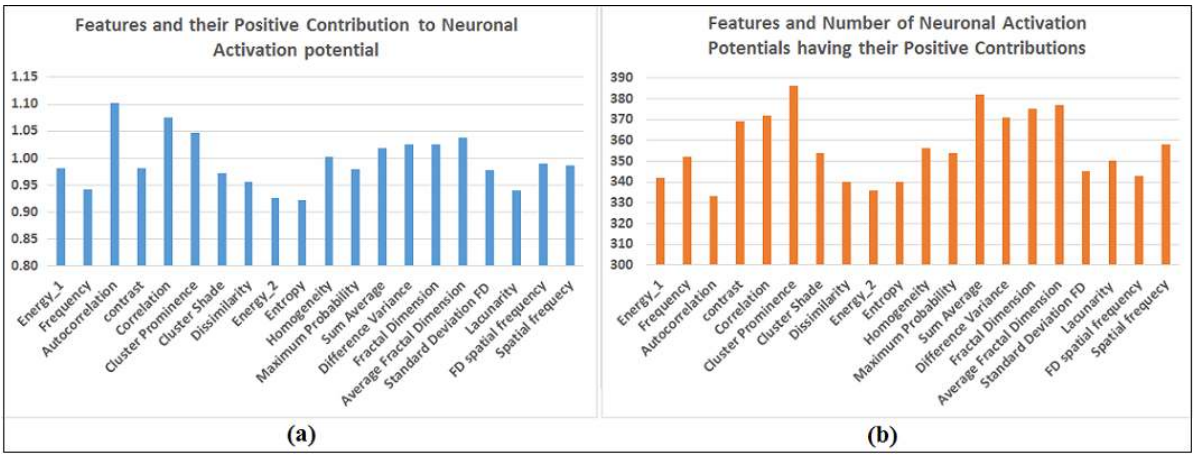


Figure 2.3: Features and their contributions to the neuronal activation potential. (a) shows the positive neuronal activation potential contributions by the candidate features as percentages of the total positive activation potential of all hidden neurons in L_1, while (b) shows each candidate feature and the number of neuronal activation potentials of all hidden neurons in L_1 having its positive contribution.

- The percentage of its positive contribution to the activation potential of all neurons in the hidden layer being analyzed.

- The net positive contribution of the feature to the activation potential of all neurons in the hidden layer being analyzed.

Table 2.1 and Figure 2.3 provide a summary of the results of the neuronal activation potential analysis. We selected the initial set of features used in diffusing the latent fingerprint dataset using their total positive neuronal activation potential contribution (PNAPC) shown in column 5 of Table 2.1. Given that the higher the activation potential of a feature, the more likely it will contribute to learning, we assumed that features with high PNAPC are more likely to influence the participation of the neurons in the input reconstruction and subsequent reduction of the reconstruction error. Based on the minimum description length principle, we used algorithm 1 to optimize the length of feature set used to diffuse the latent fingerprint dataset.

1	2	3	4	5	6	7	8
Feature	**Type**	**Positive Count**	**Negative Count**	**PNAPC**	**PNAPC** (%)	**TNAP** (%)	**Selected**
Energy-1	Gabor	342	458	46.46	0.98	1.13	✓
Frequency	Gabor	352	448	44.63	0.94	1.13	✗
Autocorrelation	GLCM	333	467	52.22	1.10	1.26	✓
contrast	GLCM	369	431	46.45	0.98	0.73	✓
Correlation	GLCM	372	428	50.95	1.07	0.58	✓
Cluster Prominence	GLCM	386	414	49.58	1.05	0.57	✓
Cluster Shade	GLCM	354	446	46.07	0.97	1.15	✗
Dissimilarity	GLCM	340	460	45.34	0.96	1.13	✗
Energy-2	GLCM	336	464	43.88	0.93	1.12	✗
Entropy	GLCM	340	460	43.74	0.92	1.11	✗
Homogeneity	GLCM	356	444	47.53	1.00	1.14	✓
Maximum Probability	GLCM	354	446	46.42	0.98	1.14	✓
Sum of Average	GLCM	382	418	48.27	1.02	0.57	✓
Difference of Variance	GLCM	371	429	48.63	1.03	0.57	✓
Fractal Dimension (FD)	Fractal Dimension	375	425	48.61	1.03	0.57	✓
Average FD	Fractal Dimension	377	423	49.18	1.04	0.57	✓
Standard Deviation FD	Fractal Dimension	345	455	46.34	0.98	1.14	✓
Lacunarity	Fractal Dimension	350	450	44.57	0.94	1.13	✗
FD spatial frequency	Fractal Dimension	343	457	46.88	0.99	1.13	✓
Spatial frequency	Spatial Frequency	358	442	46.74	0.99	1.14	✓

Table 2.1: Features and their contributions to the neuronal activation potential. Column 3 in the table shows the number of neuron in the first hidden layer L_1 where the feature contributed positively to the neuronal activation potential. Column 4 shows the number of neuron in L_1 where the feature contributed negatively to the neuronal activation potential. The total positive neuronal activation potential contribution (PNAPC) by each feature over all neurons in L_1 are shown in column five. PNAPC is shown as percentages in column 6. The total (positive and negative) neuronal activation potential contributions (TNAP) over all neurons in L_1 are shown in column 7. Column 8 indicates whether the feature was selected as part of the feature set that is further refined using Algorithm 1. PNAPC threshold of 0.98% was used in selecting the candidate features.

Minimum Description Length Feature Selection

Selecting the minimal (optimal) set of features is important because using more features than necessary increases system complexity and may not lead to better performance [26]. Using many features to diffuse the dataset implies using a complex model to approximate the training dataset. The minimum description length principle (MLDP) states that a simple model is better that a complex model [108]. Given that the patch dataset is inherently noisy, a complex model will likely over-fit the training data due to its sensitivity to noise, leading to degradation of the performance of the model on unseen examples. We performed experiments to determine a set of features and length of features which when used to diffuse the dataset yielded the best classifier performance. Starting with the initial set *F* of features selected via deep learning, we ran several experiments each time diffusing the dataset with feature vector of length *k* using Algorithm 1, while keeping other model training parameters constant. In each step, we picked *k* features where $k \in S$ and *S* is the set of all subsets (powerset) of $S = 2^{|F|}$. After each step, we removed all subsets that were used in the preceding step from *S* before proceeding with the next step. We assessed the goodness of each feature vector subset s by examining the mean square reconstruction error (MSRE) and classification error cost (EC) obtained when the model was trained with s. MSRE defined as:

$$MSRE = \frac{1}{V}\sum_{i=1}^{V} |\upsilon_i - \hat{\upsilon_i}|^2 \tag{2.5}$$

where υ_i refers to a training sample, $\hat{\upsilon_i}$ refers to its learned representation and *V* is the number of training samples. EC is defined as:

$$EC = \frac{1}{V}\sum_{i=1}^{V} |c_i - \hat{c_i}|^2 \tag{2.6}$$

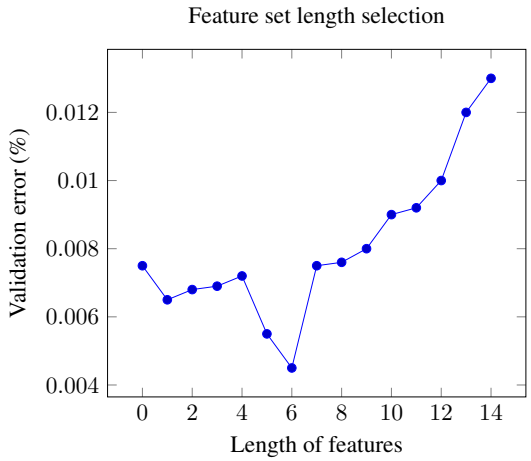


Figure 2.4: Plot showing length of features used in dataset diffusion vs. validation error. It can be seen from the plot that the best performance was obtained with feature set of length 6. The length of the set of featured selected via deep learning was 14. This plot shows that using 6 of the 14 features was better than using all the 14 selected features in diffusing the latent fingerprint patch dataset.

where c_i is the expected class of sample υ_i and $\hat{\upsilon_i}$ is the predicted class. The feature vector subset with the minimum costs and minimum length was chosen to diffuse the patch dataset. If no subset met both cost and length criteria, we chose the one with the minimum costs. Figure 2.4 shows the plot of validation error against length of feature set. The minimum validation error was obtained with a feature subset of length 6. . After diffusing the patch dataset with the selected feature subset, each image patch previously represented with a vector of 64 elements was now represented with 70 element (64+6). To create a new square matrix representation of each image patch, we introduced a padding with 11 zeros to get 81 = 9x9. Doing so did not degrade the performance of the model being trained because zero-padding does not add any information to the padded data [24].

Algorithm 1 Minimum Description Length Feature Selection.

procedure SELECTFEATURE(W) ▷ Initial Set of Selected Features via Deep Learning

  $r \leftarrow 2$ ▷ Number of features to start with

  $f \leftarrow \{$**first two features from** $W\}$

  $S_t \leftarrow \emptyset$

  while $S_{FD} \neq \emptyset$ **do** ▷ We are done if W is empty

    $S_t \leftarrow S_t \cup f$ ▷ Select first r elements from W

    $S_p \leftarrow$ PowerSet(S_t) ▷ Compute powerset of S_t

    $S_d \leftarrow \{d | d \in S_p$ and $k \in f$ and $k \notin d\}$

    $S_s \leftarrow S_p \setminus S_d$ ▷ Remove all sets in S_p containing none of the elements in f

    for $s \in S_s$ **do**

      $\hat{X} \leftarrow$ **diffuse the training dataset** X **with** s

      Train model with $\hat{X}$

      $MSRE_s \leftarrow MSRE$ obtained using $\hat{X}$ ▷ Pre-training phase mean square reconstruction error

      $EC_s \leftarrow EC$ obtained using $\hat{X}$ ▷ Fine-tuning phase error cost

      Remove s **from** S_s

    $r \leftarrow r + 1$ ▷ Add the next feature from W

    $f \leftarrow \{next\ feature\ from\ W\}$

  return the s **with min(Length), min(MSRE) and min(EC)** ▷ Return the optimal feature subset

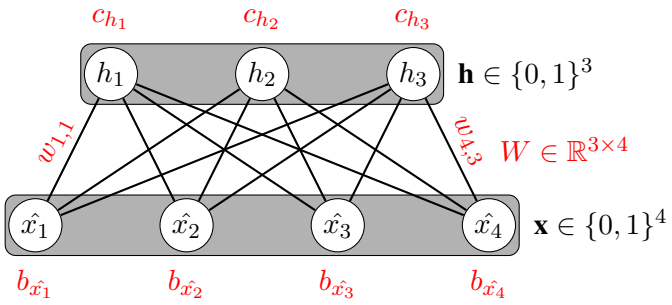


Figure 2.5: Graphical depiction of RBM with binary visible and hidden units. $\hat{x}_i, i = 1, \ldots, 4$, are the visible units while $h_k, k = 1, \ldots, 3$, are the hidden units. $b_{\hat{x}_i}, i = 1, \ldots, 4$, are the biases for the visible units and $c_{h_k}, k = 1, \ldots, 3$, are the biases for the hidden units.

2.2.2 Restricted Boltzmann Machine

A Restricted Boltzmann Machine is a stochastic neural network that consists of visible layer, hidden layer and a bias unit [71]. A sample RBM with binary visible and hidden units is shown in Figure 2.5. The energy function E_f of RBM is linear in its free parameters and is defined as [71]:

$$E_f(\hat{x}, h) = -\sum_i b_i \hat{x}_i - \sum_j c_j h_j - \sum_i \sum_j \hat{x}_i w_{i,j} h_j \tag{2.7}$$

where $\hat{x}$ and h represent the visible and hidden units respectively, W represents the weights connecting $\hat{x}$ and h, while b and c are biases of the visible and hidden units, respectively. The E_f can be written in matrix form as:

$$E_f(\hat{x}, h) = -b^T \hat{x} - c^T h - \hat{x}^T W h \tag{2.8}$$

The probability distributions over visible or hidden vectors are defined in terms of the energy function [71]:

$$P(\hat{x}, h) = \frac{1}{\omega} e^{-E_f(\hat{x}, h)} \tag{2.9}$$

where ω is a partition function that ensures the probability distribution of over all possible configurations of the hidden or visible vectors sum to 1. The marginal probability of a visible vector

$P(\hat{x})$ is the sum over all possible hidden layer configurations [71] and is defined as:

$$P(\hat{x}) = \frac{1}{\omega} \sum_{h} e^{-E_f(\hat{x},h)} \tag{2.10}$$

RBM has no intra-layer connections and given the visible unit activations, the hidden unit activations are mutually independent. Also, the visible unit activations are mutually independent given the hidden unit activations [35]. The conditional probability of a configuration of the visible units is given by

$$P(\hat{x}|h) = \prod_{i=1}^{n} P(\hat{x}_i|h), \tag{2.11}$$

where *n* is the number of visible units. The conditional probability of a configuration of hidden units given visible units is

$$P(h|\hat{x}) = \prod_{j=1}^{m} P(h_j|\hat{x}), \tag{2.12}$$

where *m* is the number of hidden units.

$$P(h_j = 1|\hat{x}) = \sigma\left(b_j + \sum_{i=1}^{n} w_{i,j}\hat{x}_i\right) \tag{2.13}$$

and

$$P(\hat{x}_i = 1|h) = \sigma\left(c_i + \sum_{j=1}^{m} w_{i,j}h_j\right) \tag{2.14}$$

where c_i is the *i-th* hidden unit bias, b_j is the *j-th* visible unit bias, $w_{i,j}$ is the weight connecting the *i-th* visible unit and *j-th* hidden unit, and σ is the logistic sigmoid.

2.2.3 Features for complex texture characterization

Texture is defined as a regular repetition of an element or pattern on a surface. For images, texture property represents the surface and structure, and may be regarded as a similarity grouping

[110]. We computed three types of features: Gabor, gray level co-occurrence matrix and fractal dimension. The features have capability of discriminating between fingerprint and non-fingerprint patches. We used deep learning to select the best set of features with minimum length for diffusing the latent fingerprint patch dataset.

Energy and Frequency Features (Features 1-2)

Energy and frequency features are used for latent fingerprint image segmentation. For five scales and eight orientations, we computed the mean energy and mean frequency for an image patch.

Gray Level co-occurrence matrix (features 3-14)

Gray-Level Co-occurrence Matrix (GLCM) is one of the earliest methods used for texture feature analysis and extraction. It was proposed by Haralick et al. in 1973 [67]. We use co-occurrence matrix to measure the texture of each patch image. A Gray Level Co-occurrence Matrix - (GLCM) is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image. We define a co-occurrence matrix *C* is defined over an $a \times b$ image patch I_p, parameterized by an offset $(\Delta x, \Delta y)$, as:

$$C(i,j) = \sum_{x=1}^{n} \sum_{y=1}^{m} \begin{cases} 1, & \text{if } I_p(x,y) = i,\ I_p(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise} \end{cases}$$

where *i* and *j* are the image pixel intensity values, *x* and *y* are the spatial positions in the image patch I_p and the offset (Δ_x, Δ_y) depends on the direction θ, and the distance δ for which the matrix is computed.

Given that spatial distribution of gray values is one of the defining qualities of texture, we used statistical methods to analyze the spatial distribution of gray values by computing local features at each point in the image patch. GLCM estimates image properties related to second-order (two pixels) statistics by considering the relationship among groups of two pixels. We calculated gray-level co-occurrences matrices for each image patch using four different offsets ,$\{[0\ 1], [-1\ 1], [-1\ 0], [-1\ -1]\}$ that are defined as one neighboring pixel in the four directions $0^o; 45^o; 90^o$ and 135^o, and computing the average over the four directions. This ensures that we capture all possible texture patterns in an image patch. Each element (i, j) of the matrix is the number of occurrences of the pair of pixel with value i and another pixel with value j at a distance d relative to each other. We examined the spatial relationship between two neighboring pixels with different offsets and angles $(0^o; 45^o; 90^o$ and $135^o)$ and extracted the GLCM features shown in Table 2.2 from the patch dataset [67]. Figure 2.6 shows plots of some GLCM features highlighting their potential for discriminating between fingerprint and non-fingerprint patches.

Feature Name	#	Description	Equation
Autocorrelation	3	Autocorrelation measures oscillatory behavior in signals. For latent fingerprint patches, it can help verify the presence of ridges and valleys.	$F(k,r) = \frac{\sum_{i,j=0}^{N-1} P(i,j)P(i+k,j+r)\varphi}{\frac{1}{N^2}\sum_{i,j=0}^{N-1} P(i,j)^2}$, (15) $\varphi = \frac{1}{(N-k)(N-r)}$ (16)
Contrast	4	This feature measures the local variations in intensity between a pixel and its neighbors. The larger the contrast the larger the intensity differences in the GLCM.	$\sum_{i,j=0}^{N-1} (i-j)^2 P(i,j),$ (17)
Correlation	5	This is a measure of how correlated a pixel is to its neighbor.	$\sum_{i,j=0}^{N-1} P(i,j)^2 \left[\frac{(i-\vartheta_i)(j-\vartheta_j)}{\sqrt{\sigma_i^2\sigma_j^2}}\right]$, (18)
Cluster Prominence	6	This is a measure of asymmetry in an image. For fingerprint patches, a high value of cluster prominence indicates small variations in gray-scale values.	$\sum_{i,j=0}^{N-1} P(i,j)(i+j-\vartheta_i-\vartheta_j)^4,$ (19)
Cluster Shade	7	A measure of the skewness of an image. A high value of cluster shade indicated that the image is asymmetric.	$\sum_{i,j=0}^{N-1} P(i,j)(i+j-\vartheta_i-\vartheta_j)^3,$ (20)
Dissimilarity	8	Sum of difference in intensity values between a reference pixel and its neighbors.	$\sum_{i,j=0}^{N-1} P(i,j)\lvert i-j\rvert,$ (21)
Energy	9	A measure of uniformity in the image patch	$\sum_{i,j=0}^{N-1} P(i,j)^2,$ (22)
Entropy	10	Entropy measures the randomness in the image. When all the elements in the GCLM are similar, entropy is high and when they are dissimilar, entropy is small.	$-\sum_{i,j=0}^{N-1} P(i,j) log P(i,j),$ (23)
Homogeneity	11	This feature is a measures how close the distribution of elements in the GCLM.	$\sum_{i,j=0}^{N-1} \frac{P(i,j)}{1+(i-j)^2},$ (24)
Maximum Probability	12	Measures the strongest response of the co-occurrence matrix P	$MAX_{i,j}(P(i,j))$
Sum of Average	13	Sum of the mean (ϑ_i) based on the reference pixel i and mean (ϑ_j)based on neighboring pixels j	$\vartheta_i = \sum_{i,j=0}^{N-1} iP(i,j), \vartheta_j = \sum_{i,j=0}^{N-1} jP(i,j$ (25)
Difference of Variance	14	This feature provides a measure of the dispersion of the difference between the reference and the neighbour pixels in a given image patch. It is given by $\lvert\sigma_i^2 - \sigma_j^2\rvert$	$\sigma_i^2 = \sum_{i,j=0}^{N-1} P(i,j)(i-\vartheta_i)^2,$ (26) $\sigma_j^2 = \sum_{i,j=0}^{N-1} P(i,j)(j-\vartheta_j)^2,$ (27)

Table 2.2: GLCM Features used in our experiment. The features were computed based on 4 directions of analysis $(0^o; 45^o; 90^o$ and $135^o)$, i and j are the gray level values in the image patch.

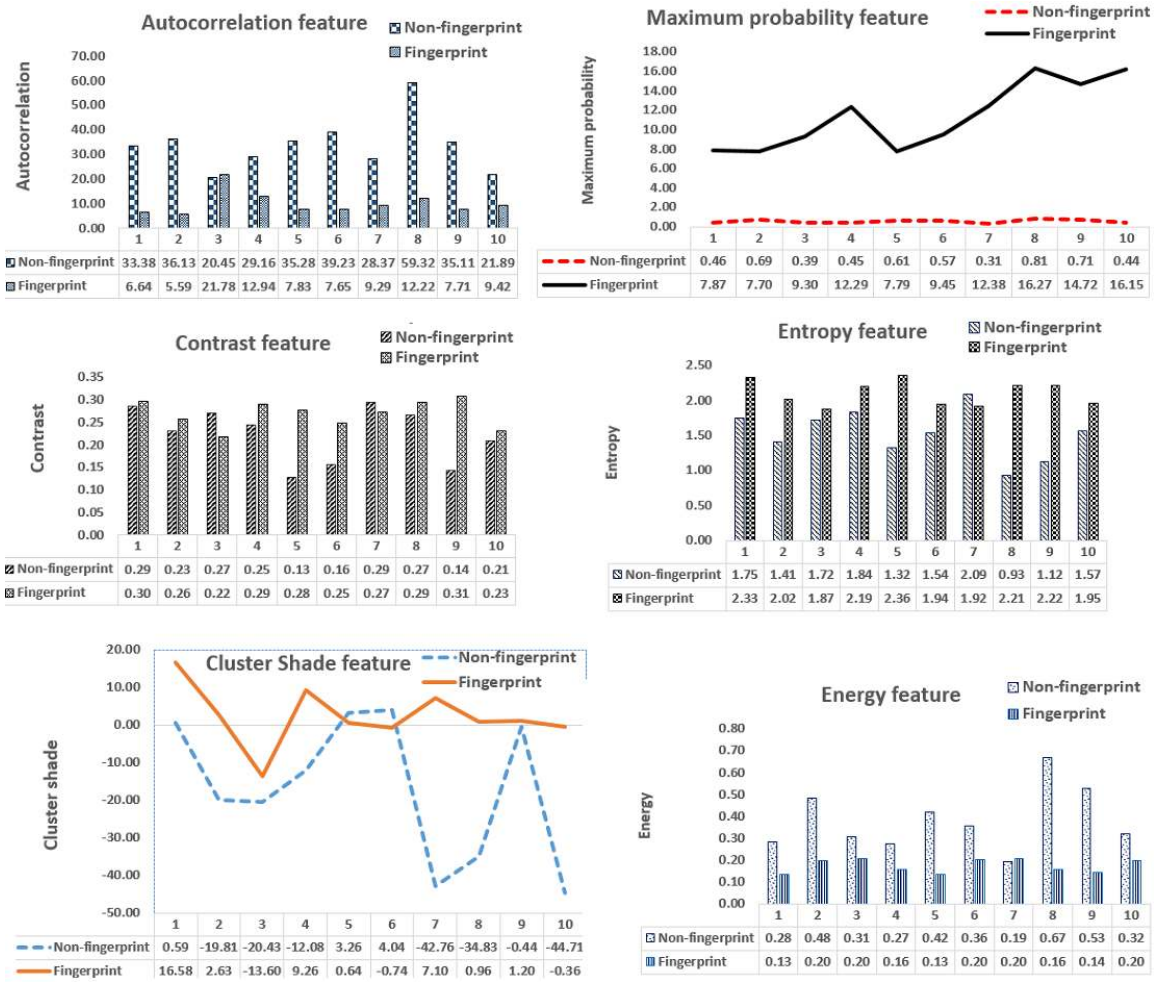


Figure 2.6: Examples of GLCM features for 10 fingerprint and 10 non-fingerprint patches from NIST SD27 latent fingerprint images. As can be seen from the figures, GLCM features can be used to discriminate between fingerprint and non-fingerprint image patches.

Fractal Dimension Features (Features 15-20)

- ***Fractal Dimension (Feature 15)*** : Fractal dimension is an index used to characterize texture patterns by quantifying their complexity as a ratio of the change in detail to the change in the scale used. It was defined by Mandelbrot [98] and was first used in texture analysis by Keller et al. [80]. Fractal dimension offers a quantitative way to describe and characterize the complexity of image texture composition [85]. It can be used to discriminate between fingerprint and non-

fingerprint parts of the image. As can be seen from Figure 2.7, non-fingerprint patches show larger fractal dimension values than fingerprint patches. The intensity values of the pixels in fingerprint patches tend to be widely separated due to the ridges and valleys and, therefore, yield lower values for fractal dimension, while the intensity values for pixels in non-fingerprint patches tend to be close to each, resulting in a high value fractal dimension.

We computed the fractal dimension of an image patch *P* using a variant of differential box-counting (DBC) algorithm [13, 119]. We consider *P* as a *3-D* spatial surface with *(x,y)* axis as the spatial coordinates and z axis for the gray level of the pixels. Using the same strategy as in *DBC*, we partition the $N \times N$ matrix representing *P* into non-overlapping $d \times d$ blocks where $d \in [1, N]$. Each block has a column of boxes of size $d \times d \times h$, where *h* is the height defined by the relationship $h = \frac{\mathcal{T}d}{N}$, where $\mathcal{T}$ is the total gray levels in *P*, and *d* is an integer. Let $\mathcal{T}_{min}$ and $\mathcal{T}_{max}$ be the minimum and maximum gray levels in grid *(i, j)*, respectively. The number of boxes covering block *(i, j)* is given by:

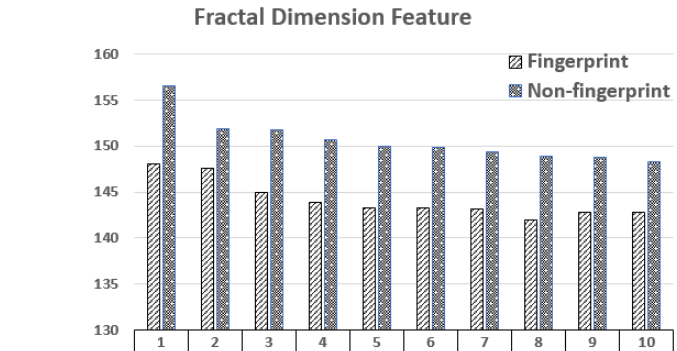
$$n_d(i,j) = floor[\frac{\mathcal{T}_{max} - \mathcal{T}_{min}}{r}] + 1, \tag{2.15}$$

where $r = 2, \ldots, N-1$, is the scaling factor and for each block *r* = *d*. The number of boxes covering all $d \times d$ blocks is:

$$N_d = \sum_{i,j} n_d(i,j) \tag{2.16}$$

We compute the values N_d for all $d \in [1, N]$. The fractal dimension of each pixel in *P* is by given by the slope of a plot of the logarithm of the minimum box number as a function of the logarithm of the box size. We obtain a fractal dimension image patch P' represented by an $M \times N$ matrix whose entry *(i, j)* is the fractal dimension FD_{ij} of the pixel at *(i, j)* in *P*.

$$FD_P = \sum_{i=1,j=1}^{MN} FD_{ij} \tag{2.17}$$



	1	2	3	4	5	6	7	8	9	10
Fingerprint	148.06	147.58	144.99	143.89	143.32	143.28	143.12	141.92	142.84	142.75
Non-fingerprint	156.57	151.86	151.76	150.73	150.01	149.82	149.34	148.96	148.83	148.27

Figure 2.7: Examples of the fractal dimension feature for 10 fingerprint and 10 non-fingerprint patches patches from NIST SD27 latent fingerprint images.The chart highlights the discriminative potential of fractal dimension features for fingerprint and non-fingerprint image patches.

We implemented a variant of the *DBC* algorithm [13, 119], to compute the following statistical features from the fractal dimension image P'.

• ***Average Fractal Dimension (Feature 16)*** :

$$FD_{avg} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} FD_{ij} \tag{2.18}$$

• ***Standard Deviation of Fractal Dimension (Feature 17):*** The standard deviation of the gray levels in an image provides a degree of image dispersion and offers a quantitative description of variation in the intensity of the image plane. Therefore,

$$FD_{std} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} (FD_{ij} - FD_{avg})^2, \tag{2.19}$$

• ***Spatial Frequency and FD Spatial Frequency***

(Features 18 & 19): This refers to the frequency of change per unit distance across fractal dimension

(FD) processed image. We compute it using the formula for (spatial domain) spatial frequency [91]. Given an $N \times N$ FD processed image patch P', let *G(x,y)* be the FD value of the pixel at location *(x,y)* in P'. The row frequency R_f and column frequency C_f are given by

$$R_f = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=1}^{N-1} [G(x,y) - G(x,y-1)]^2} \tag{2.20}$$

$$C_f = \sqrt{\frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=1}^{N-1} [G(x,y) - G(x-1,y)]^2} \tag{2.21}$$

The FD spatial frequency FD_{sf} of P' is defined as

$$FD_{sf} = \sqrt{R_f^2 + C_f^2} \tag{2.22}$$

From signal processing perspective, equations (2.20) and (2.21) favor high frequencies and yield values indicative of patches with fingerprint.

Lacunarity (Feature 20)

Lacunarity is a second-order statistic that provides a measure of how patterns fill space. Patterns that have more or larger gaps have higher lacunarity. It also quantifies rotational invariance and heterogeneity. A spatial pattern that has a high lacunarity has a high variability of gaps in the pattern, and indicates a more heterogeneous texture [21]. Lacunarity (FD_{lac}) is defined in terms of the ratio of variance over mean value [13].

$$FD_{lac} = \frac{\frac{1}{MN}(\sum_{i=1}^{M-1} \sum_{j=1}^{N-1} P(i,j)^2)}{\{\frac{1}{MN} \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} P(i,j)\}^2} - 1 \tag{2.23}$$

where *M* and *N* are the sizes of the fractal dimension image patch *P*.

2.3 Experiments and Results

We implemented our algorithms in Matlab R2014a running on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. Our implementation relied on NNBox, a Matlab toolbox for neural networks. The implementation uses backpropagation, contrastive divergence, Gibbs sampling, and hidden units sparsity based optimization techniques.

2.3.1 Databases

NIST SD27 was used for training , validating and testing the model. For matching evaluation, we used a background database consisting of rolled fingerprint images in NIST SD27, NIST SD4 and synthetic fingerprint images. Tables 2.3 and 2.4 show the protocols used in the experiments. In the evaluation of matching performance experiment, synthetic images were used to boost the size of the background database due to the unavailability of NIST SD14 database [6]. Synthetic fingerprints have been shown to be very useful for training and testing purposes, and have been used for technology evaluations [2]. As can be seen from the sample images from the background database shown in Figure 2.8, synthetic fingerprints generated using SFinGe (Synthetic Fingerprint Generator) [10], look like real fingerprints.

Experiment	Images Used	Image Size	Patches Sampled	Patch Size
Training	50	800x768	132,000	8x8
Validation	50	800x768	48,000	8x8
Testing	50	800x768	70,000	8x8
Architecture & Hyper-Parameter Selection	54	800x768	100,000	8x8
Model Stability Analysis	54	800x768	150,000	8x8
Total	258		500,000	8x8

Table 2.3: Experimental protocol showing the number of images used for model training, validation, and testing, architecture and hyper-parameter selection, and model stability analysis. Patches were sampled from latent fingerprint images in NIST SD27.

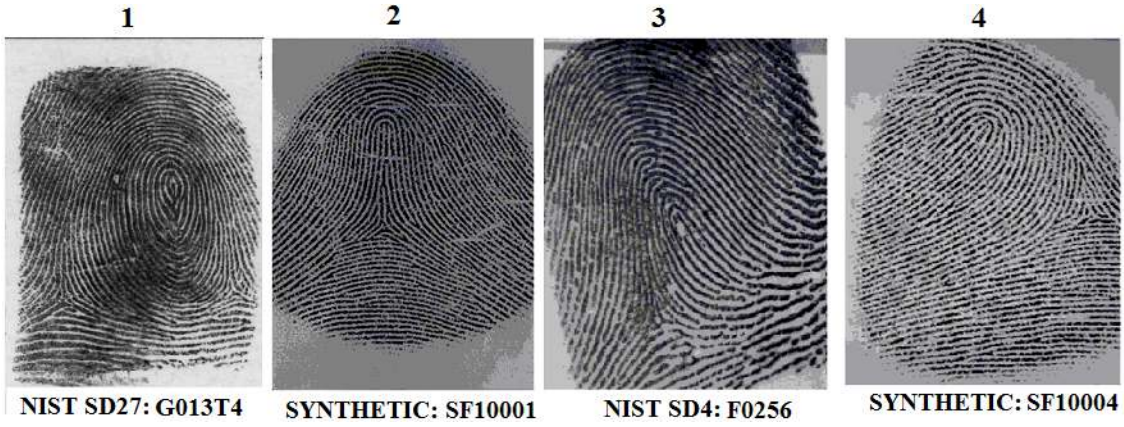


Figure 2.8: Sample fingerprints from the background database used for matching performance evaluation. The first and third fingerprints are from NIST SD27 (rolled fingerprint) and NIST SD4, respectively. The second and fourth fingerprints are synthetic fingerprints.

2.3.2 Ground-Truth Dataset

There is no existing patch based latent fingerprint ground-truth dataset. We built the ground-truth dataset by extracting 8x8 non-overlapping image patches from 50 Good, 50 Bad, and 50 Ugly 800x768 latent fingerprint images from the NIST SD27 database. For each latent fingerprint

Number of Images Used	**Image Size**
Segmented Fingerprints: 258 (NIST SD27)	$\leq$ 380x448
Background database: 257 (NIST SD27) +	800x768
2,000 (NIST SD4) +	512x512
27,000 (Synthetic fingerprints)	416x560
Total = **29,257**	

Table 2.4: Experimental protocol showing the number of images used for the evaluation of matching performance experiment.

image, we manually marked the regions containing fingerprints using bounding polygons, as the regions-of-interest (ROIs). We split a latent fingerprint into 8x8 non-overlapping patches (blocks). A patch is labeled a fingerprint patch if it overlaps with the ROI polygon and non-fingerprint otherwise. We consider a patch to be overlapped with the ROI polygon if it lies within the polygon or at least 25% of its pixels are inside the polygon.

2.3.3 Choice Architecture and Hyper-Parameters

Experiment for architecture and parameter selection was done with 100,000 8x8 patches (75,000 for training, and 25,000 for validation). We tried five different networks (with different number of layers and neurons in each layer) in the pre-training phase. The architecture that gave the best performance in terms of input reconstruction, training and validation errors, was chosen. We kept the fine-tuning layers constant since varying them did not make much difference in the final outcome. The hyper-parameters used in the proposed network were selected based on the performance of the network on the validation set. The architectures and their performance on the training and validation datasets are shown in Table 2.5. The chosen architecture is highlighted in

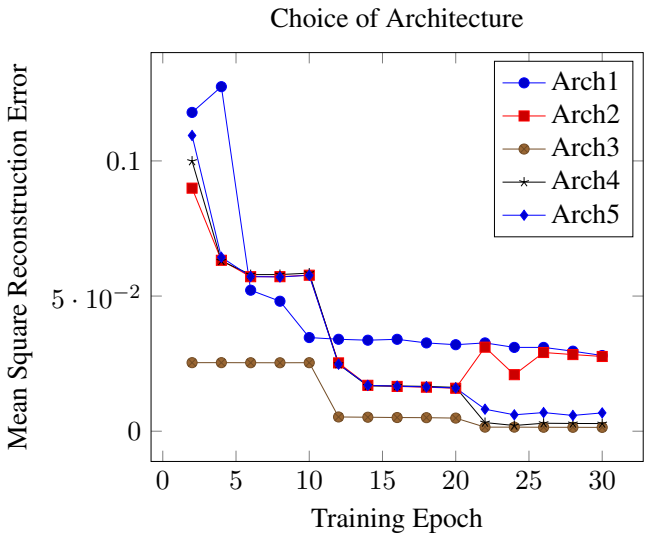
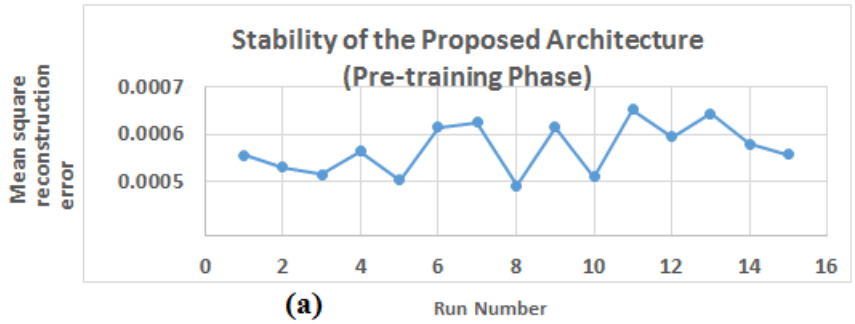


Figure 2.9: Plot showing different architectures and mean square reconstruction error. Arch3 gave the best performance.

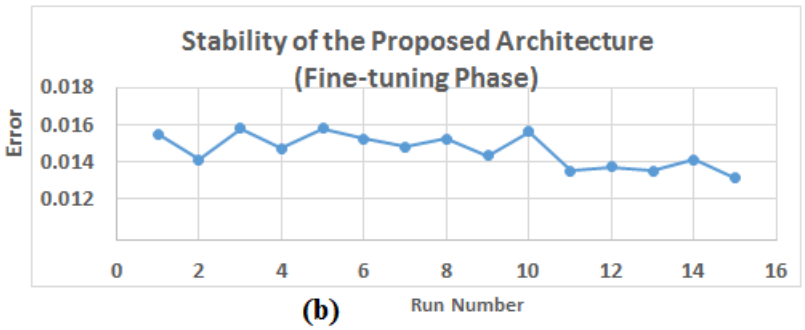
bold. Figure 2.9 shows a plot of the mean square reconstruction error against training epoch for the different architectures.

Architecture	Pre-training layers	Minimum MSRE	Maximum MSRE	Minimum Error Cost	Maximum Error Cost
Arch1	1	0.0280	0.1180	0.5852	1.041
Arch2	2	0.0276	0.0899	0.3468	0.8825
Arch3	**3**	**0.0014**	**0.0253**	**0.0007**	**0.0032**
Arch4	4	0.0028	0.0998	0.0088	0.0135
Arch5	5	0.0068	0.1094	0.0093	0.0254

Table 2.5: Five candidate architectures and model performance. The difference between the architectures is the number of pre-training layers. The architecture with 3 pre-training layers gave the best performance in terms of reconstruction and validation errors, and it was used for the feature selection and segmentation tasks in this chapter. As can be seen from the table, both reconstruction and validation errors improved as we added more pre-training layers. The gains started to fade after 3 layers. This result is consistent with the observation in [52] that unsupervised pre-training actually helps deep neural networks but after a certain depth, the benefit starts to disappear.



(a)



(b)

Figure 2.10: Network Stability: (a) shows that the mean square reconstruction error (MSRE) at convergence for the 15 runs are close, with a standard deviation of 0.0009. Similarly, (b) shows that the error during the fine-tuning phase for the 15 runs were close with a standard deviation of 2.56E-05. These results are indicative of the stability of the network. Each data point in the graph is the error at convergence of the matching run number.

2.3.4 Stability of the Architecture

To investigate the stability of the proposed architecture, we randomly sampled 150,000 8x8 patches from the ground-truth dataset. We performed 15 runs of network training and for each of the 15 runs, we used 50,000 patches randomly sampled from the 150,000 patches. All the model parameters (number of epochs, number of iterations, etc.) remained unchanged across the runs. The mean square reconstruction error (MSRE) and mean error cost at convergence, as well as the standard deviation for the 15 runs are shown in Table 2.6. Plots of the reconstruction errors and error cost at convergence against number of runs are shown in Figure 2.10. These results show that

Run #	MSRE	Error Cost	MDR	FDR
1	0.0155	5.569E-04	3.120E-04	0.00
2	0.0141	5.306E-04	4.150E-04	0.00
3	0.0158	5.160E-04	1.810E-04	0.00
4	0.0147	5.638E-04	3.170E-04	0.00
5	0.0158	5.045E-04	2.080E-04	0.00
6	0.0152	6.145E-04	1.090E-04	0.00
7	0.0148	6.245E-04	3.650E-04	0.00
8	0.0152	4.915E-04	2.140E-04	0.00
9	0.0143	6.145E-04	3.510E-04	0.00
10	0.0156	5.105E-04	1.560E-04	0.00
11	0.0135	6.515E-04	2.610E-04	0.00
12	0.0137	5.955E-04	4.260E-04	0.00
13	0.0135	6.445E-04	3.270E-04	0.00
14	0.0141	5.805E-04	2.590E-04	0.00
15	0.0131	6.845E-04	2.150E-04	0.00
Mean	0.0146	5.3436E-04	2.8660E-04	0.00
Standard Deviation	0.0009	2.5581E-05	9.4055E-05	0.00

Table 2.6: Network Stability: The mean square reconstruction error (MSRE) at the pre-training phase, error at fine-tuning phase, MDR, and FDR for the 15 different runs are close. The mean and standard deviation indicate stability across the 15 runs.

the proposed model is stable.

2.3.5 Training, Validation and Testing

The training, validation and testing of the model was done with 250,000 patches consisting of 90,000, 80,000 and 80,000 8x8 patches from the Good, Bad and Ugly ground-truth dataset categories, respectively. We selected more patches (90,000) from the Good category to have more of the good quality fingerprint patches. For training and validation, 180,000 patches were randomly

sampled from the 250,000 patches. 132,000 patches (73%) were used for training while 48,000 patches (27%) were used for validation. There was no noticeable performance gain when the model was trained with more than 132,000 patches despite taking longer to converge. The trained model was tested with the remaining 70,000 patches. There was no overlap between the training, validation and test datasets. Table 2.7 shows the confusion matrices for training, validation and testing.

		Predicted Patch Class (Training)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**23,663**	**13**
	Non-Fingerprint	**0**	**108,324**

		Predicted Patch Class (Validation)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**12,941**	**159**
	Non-Fingerprint	**1**	**34,899**

		Predicted Patch Class (Testing)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**16,858**	**272**
	Non-Fingerprint	**4**	**52,866**

Table 2.7: NIST SD27 - Confusion matrix for training, validation and testing.

2.3.6 Pattern Derivation Using Dataset Diffusion

Given a latent fingerprint dataset $X = \{x_1, x_2, \cdots, x_k\}$, we derive new patterns $X^p = \{p_1, p_2, \cdots, p_w\}$ by computing statistical features from *X*. The new patterns include fractal dimen-

sion features, lacunarity, and spatial frequency. We extend X with X^p to form a new dataset $\hat{X} = \{x_1, x_2, \cdots, x_k, p_1, p_2, \cdots, p_w\}$ and used $\hat{X}$ to train and validate the network. We tried to improve the performance of the model without using dataset diffusion by experimenting with various data augmentation techniques such as label preserving transformation and oversampling/undersampling of the minority/majority samples to balance the dataset. We also tried other learning techniques such as one class learning. None of those techniques yielded the desired segmentation results. In subsequent experiments, we observed that diffusing the dataset yielded a trained model that has better generalization on unseen examples. A comparison of the results obtained with and without dataset diffusion is shown in Figure 2.11. As can be seen from Figure 2.11, when the training dataset was *extended* with the selected features, there was a huge drop in both the error cost during fine-tuning and the classification error during training. The reconstruction error almost remained the same in both cases.

2.3.7 Performance Evaluation and Metrics

We used the following metrics to evaluate the performance of our network.

- **Missed Detection Rate (*MDR*):** This is the percentage of fingerprint patches classified as non-fingerprint patches.

$$MDR = \frac{FN}{TP + FN} \tag{2.24}$$

  where *FN* is the number of false negatives and *TP* is the number of true positives.

- **False Detection Rate (*FDR*):** This is the percentage of non-fingerprint patches classified as

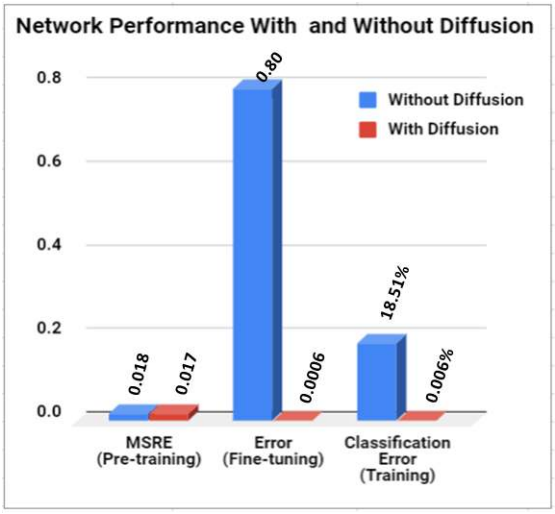


Figure 2.11: Impact of Data Diffusion on the performance of our model during training. During the pre-training phase, the network achieves lower mean square reconstruction error (MSRE) when the dataset is diffused compared to when it is not diffused. Also, by diffusing the dataset we get faster convergence and lower error cost during the fine-tuning phase than when the dataset is not diffused. Through experiments, we found that applying data diffusion in the data-space (before deep learning) instead of in the feature-space (after deep learning), improved the performance of our model. This is consistent with the finding in [143] that data augmentation in the data-space reduces over-fitting and leads to better performance improvement than performing the augmentation in the feature-space. The MSRE (pre-training) and Error (fine-tuning) are values at 50 epochs (lower is better). The classification errors are values at iteration 50 (lower is better). This figure is better when viewed in color.

fingerprint patches. It is defined as

$$FDR = \frac{FP}{TN + FP} \tag{2.25}$$

where *FP* is the number of false positives and *TN* is the number of true negatives.

- **Segmentation Accuracy (SA):** It gives a good indication of the segmentation reliability of the model.

$$SA = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.26}$$

2.3.8 Results and Comparison with Current Algorithms

Figure 2.12 shows segmentation performance of our proposed method on sample images from the NIST SD27 database. The figure shows original latent fingerprint images and the segmented fingerprints constructed using patches that are classified as fingerprints. Table 2.8 shows the superior performance of our segmentation approach on the NIST SD27 good, bad and ugly quality latent fingerprints compared to existing algorithms.

Please note that a deep learning model trained with a subset NIST SD27 (containing images with lots of structured noise and poor quality) and later used on it in "production" mode is expected to achieve segmentation results that are better than those from another model trained on a database with cleaner and better quality images, and then deployed on NIST SD27 in production mode. Training with "ugly" data and tuning the model parameters until good performance is achieved produces a more robust model that is able to handle challenging datasets (in the same domain). This is the advantage our model has over others. There is no need to apply the "holdout" rule when the model is in production mode.

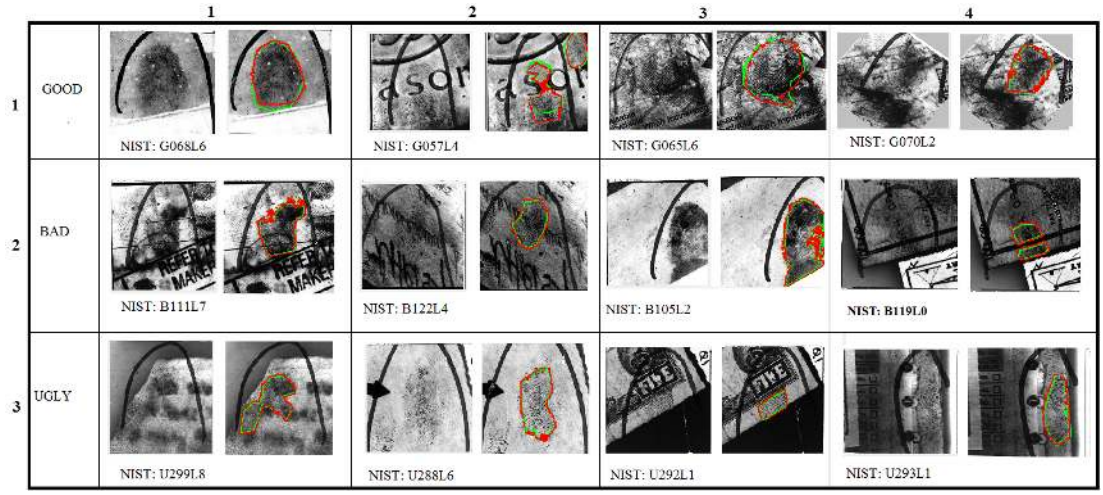


Figure 2.12: Good, Bad and Ugly latent fingerprint images and segmentation results. Each row contains the original latent fingerprint images on the left side of the columns. The right side of the columns contain the original images with the ground-truth fingerprint regions marked with green bounding polygons, and the segmented fingerprint parts marked with red bounding polygons. The multiple segmented fingerprints in row 1 column 2, and row 2 column 4, show that our segmentation algorithm is able to segment multiple fingerprint impressions on the same latent fingerprint.

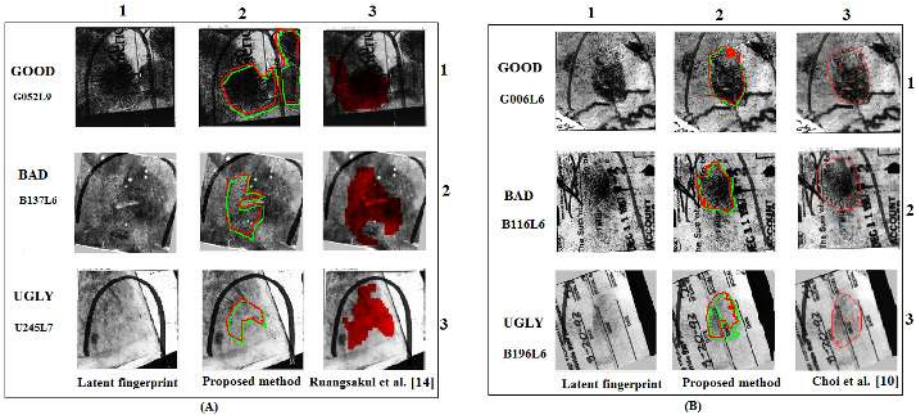


Figure 2.13: Visual comparison of the segmentation results. (A) Proposed method and that of Ruangsakul et al. [111] for NIST SD27 G052L9 (Good), B137L6 (Bad), and U245L7 (Ugly) latents. (B) Proposed method and that of Choi et al. [39] for NIST SD27 G006L6 (Good), B116L6 (Bad) and B196L6 (Bad) latents. In the segmentation results from proposed approach, the original images are shown with the ground-truth fingerprint regions marked with green bounding polygons, and the segmented fingerprint parts marked with red bounding polygons. In the other approaches, the ground-truth regions in the fingerprint images are not indicated. It should be noted that the authors in [39] referred to B196L6 as U196L6 is an error because U196L6 does not exist in NIST SD27 database. The images shown are as published by the authors. We could not show the visual comparison of the segmentation results to those of Cao et al. [31] because the images shown on their Fig. 8. "Illustration of latent fingerprint segmentation" were not named, making it hard to find the matching images in NIST SD27 for the comparison test.

Category	Author	Method	MDR (%)	FDR (%)	Average
S_1	Choi et al. [39]	Statistical	14.78	47.99	31.38
	Zhang et al. [153]	Statistical	14.10	26.13	20.12
	Arshad et al. [19]	Machine Learning	4.77	26.06	20.12
	Ezeobiejesi et al. [53]	Machine Learning (patch based)	9.22	18.70	13.96
S_2	Zhu et al. [155]	Deep Learning (patch based)	10.94	11.68	11.31
	This chapter	**Deep Learning** (patch based)	**1.14**	**0.07**	**0.61**
S_3	Nguyen et al. [104]	Deep Learning (pixel based)	2.57	16.36	9.46

Table 2.8: Comparison with other algorithms. There is no uniform protocol for evaluating latent fingerprint segmentation algorithms. For a fair comparison of the results from different approaches, we split the algorithms into two categories. Category S_1 is for segmentation methods that use NIST SD27 for both hypothesis development and testing (of statistical models), and training and testing (of machine learning models). Category S_2 is for deep learning based segmentation methods that use NIST SD27 for both training and testing. Category S_3 is for deep learning based segmentation methods that use a different database for training and NIST SD27 for testing. The proposed segmentation approach shows superior segmentation performance on the good, bad and ugly quality latent fingerprints from NIST SD27 database compared to existing algorithms. The results shown are as published by the authors.

Figure 2.13 shows a visual comparison of the segmentation results of our proposed method and two other segmentation methods that included the NIST SD27 image numbers corresponding to the authors' published segmentation results. Figure 2.13 box A shows the segmentation results of our proposed method and that of Ruangsakul et al. [111] for NIST SD27 G052L9 (Good), B137L6 (Bad), and U245L7 (Ugly) latents, while box B compares the segmentation results of the proposed

method to that of Choi et al. [39] for NIST SD27 G006L6 (Good), B116L6 (Bad) and B196L6 (Bad) latents. It should be noted that the authors in [39] referred to B196L6 as U196L6 in error because U196L6 does not exist in NIST SD27 database. As shown in Figure 2.13 box A row 1, our segmentation algorithm is able to identify and segment multiple fingerprint impressions on the same latent fingerprint image, unlike the other algorithms.

2.3.9 Evaluation of Matching Performance

We also evaluated the accuracy of the proposed latent fingerprint segmentation method by measuring the latent matching performance using a commercial matcher, Verifinger. SDK [11]. First, the latent fingerprints (ROIs) are segmented with the proposed method and minutiae are extracted using Commercial off-the-shelf (COTS) minutiae extractor. The minutiae are then used as input to Verifinger. Note that the matching experiments are done to indirectly evaluate the quality of segmentation results, and are being performed with the segmentation system already in "production mode", so the holdout rule should not necessarily be applied. The matching performance is measured using rank identification rate.

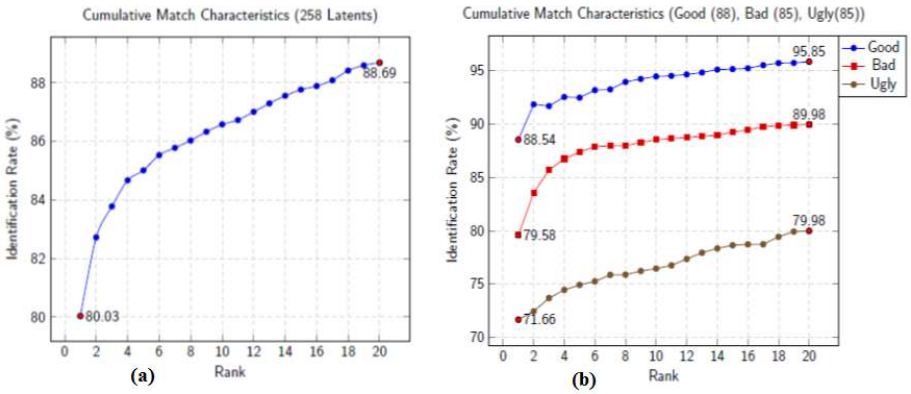


Figure 2.14: NIST SD27: (a) CMC plot of the proposed approach in matching 258 fingerprints against a test database of 2,257 rolled fingerprints. (b) CMC plots for matching the 258 fingerprints against a test database of 2,257 rolled fingerprints with the fingerprints grouped by subjective quality by latent examiners into Good (88), Bad (85), and Ugly (85). These results were obtained by running the matching tests 10 times and averaging the results.

Rank identification rate provides an estimate of the probability that a matching rolled fingerprint is identified correctly at least at rank-*k* during a search with a latent candidate. There are 258 latent fingerprints and 257 (258 less 1 duplicate) rolled fingerprints (mates of the latent fingerprints) in NIST SD27 database. We match the 258 latent fingerprints against reference fingerprint databases that include the 257 rolled fingerprints from NIST SD27. Figure 2.14(a) shows the cumulative match characteristics (CMC) curve of the proposed approach in matching 258 segmented latent fingerprints in NIST SD27 database against a database of 2,257 rolled fingerprints consisting of 2,000 fingerprint images in NIST SD4 database, and the 257 rolled images in NIST SD27 database. Figure 2.14(b) shows the CMCs of matching the three categories of latent fingerprints in the NIST SD27 database (88 Good, 85 Bad, and 85 Ugly) [74] against the test database of 2,257 rolled prints. The plot shows the rank-*k* identification rate against *k*, $k = 1, \ldots, 20$.

We show two results in Table 2.9 for the proposed method. Using the proposed method-case 1, rank-1 identification rate of 80.03% and rank-20 identification rate of 88.69% were obtained

on matching the 258 latent fingerprints in NIST SD27 against a database of 2,257 fingerprints. In proposed method-case 2, rank-1 and rank-20 identification rates of 77.05% and 84.98%, respectively, were obtained when the 258 latents were matched against a database of 29,257 fingerprints consisting of the 257 rolled images in NIST SD27, 2000 images in NIST SD4 database, and 27,000 synthetic fingerprints generated with SFinGe [10]. These results are promising when compared to the state-of-the-art rank-1 and rank-20 identification rates of 74.0% and 82.9%, respectively, reported in [75], which to the best of our knowledge, are the state-of-the-art results. Table 2.9 shows a comparison of published rank-1 and rank-20 identification rates for NIST SD27 latents. It also shows the background database and matchers used by the authors. We used 27,000 synthetic fingerprint images in place of the 27,000 fingerprints from NIST SD14 used by other authors because NIST SD14 is no longer available [6]. It should be noted that synthetic fingerprints have been shown to be very useful for training and testing purposes [42], and have been used for technology evaluations [2].

Category	Author	Background DB Size	Matcher	Rank-1	Rank-20
M_1	Choi et al. [39]	31,997 (257 from NIST SD27, 4,739 from WVU, 27,000 from NIST SD14)	COTS	16.28%	35.19%
	Ruangsakul et al. [111]	27,258 (257 from NIST SD27, 27,000 from NIST SD14)	Verifinger SDK 6.6	15.28%	22.5%
	Cao et al. [31]	31,997 (257 from NIST SD27, 4,739 from WVU, 27,000 from NIST SD14)	COTS	61.24%	-
	Proposed method-case 1	**2,257** (257 from NIST SD27, 2,000 from NIST SD4)*	**Verifinger SDK 10.0**	**80.03%**	**88.69%**
	Proposed method-case 2	**29,257** (257 from NIST SD27, 2,000 from NIST SD4, 27,000 Synthetic Fingerprints)	**Verifinger SDK 10.0**	**76.92%**	**84.17%**
M_2	Jain et al. [75]	29,257 (257 from NIST SD27, 2,000 from NIST SD4, 27,000 from NIST SD14)	COTS	74.0%	82.9%

Table 2.9: NIST SD27: Rank-1 and rank-20 identification rates for 258 latents. COTS stands for Commercial off-the-shelf. All the results quoted are as published by the authors. For a fair comparison of the results from different approaches, we split the algorithms into two categories M_1 , and M_2. Category M_1 is for segmentation papers that evaluate quality of segmentation by matching NIST SD27 images, while category M_2 is for matching only paper on NIST SD27. **NIST SD14 was not used in the proposed method-case 2 because it is no longer available* [6]. Results of the proposed method using synthetic fingerprints in place of NIST SD14 database are also shown in the table. It should be noted that synthetic fingerprints have been used in fingerprint matching competitions and research [2], it is therefore not out of place to use them in the absence of SD14 database. It might seem unreasonable to compare the results we obtained using a reference database augmented with synthetic fingerprints to results from other algorithms that used a reference database containing only real fingerprint images, but the comparison highlights the likely performance of our model in a realistic setting, compared to other algorithms.

2.3.10 Computational Cost

The preprocessing, segmentation and post processing of a single NIST SD27 latent fingerprint using the trained segmentation model takes about 2.45 seconds on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. The reported time is the average of 5 segmentation runs using 5 different latent fingerprints from NIST SD27. The processing time can be drastically reduced by deploying the model on a more powerful hardware.

2.4 Summary

This chapter proposes a new technique called dataset diffusion that enables a deep neural network achieve better estimation of the relation between input examples and a target class through efficient minimization of input reconstruction error. By extending latent fingerprint patch dataset with patterns derived from the dataset and using the extended dataset to train and validate a deep neural network, a trained model with better generalization to unseen examples is obtained. As demonstrated through experiments with and without dataset diffusion, there is marked improvement in both the error cost and classification accuracy of the model when dataset diffusion technique is used. Quantitative and visual segmentation results are presented and compared with the state-of-the-art segmentation results. A comparison of the quality of segmentation with respect to matching as well as comparison with the state-of-the-art matching results are also presented. The segmentation and matching evaluation results clearly demonstrate that using data diffusion technique to train a deep neural network for latent fingerprint image segmentation leads to marked improvement in both segmentation accuracy and matching performance.

Chapter 3

Latent Fingerprint Image Quality Assessment Using Deep Learning

This chapter proposes a deep learning model for latent fingerprint quality assessment that eliminates the need for manual feature markup. The first stage in our model uses deep learning to segment a latent fingerprint. Feature vectors computed from the segmented latent fingerprint are used as input to a multi-class perceptron that predicts the quality of the fingerprint. Experimental results on NIST SD27 fingerprint database show the promise of the proposed approach. NIST SD27 database is the most suitable database for this work because all the latent fingerprint images in it have quality labels assigned by latent experts. *To the best of our knowledge, no previous work [29, 117, 149] on latent fingerprint image quality assessment performs latent fingerprint region-of-interest segmentation and quality assessment in a lights-out mode (minimal involvement of latent examiners). This work requires no manual ROI and feature markups by latent examiners in the segmentation and quality assessment steps.*

The rest of this chapter is organized as follows. Section 3.1.1 presents a review of existing algorithms for latent fingerprint image quality estimation. Section 3.1.2 describes the contributions of this chapter while Section 3.2 highlights our technical approach and framework. Section 3.2.1 presents an overview of Restricted boltzman machine (RBM) used to build the deep learning model and as well as a brief description of the segmentation stage of our framework. The quality assessment stage is discussed in Section 3.2.3. Section 3.2.3 discusses the features used to train the quality estimation neural network layer. Experimental results and performance evaluation are presented in Section 3.3, while Section 3.4 contains the conclusions and future work.

3.1 Related Work and Contributions

3.1.1 Related Work

Fingerprint quality assessment has received considerable attention in the literature [14]. Some recent studies on latent fingerprint quality assessment used local image features for quality assessment while others used global image features. The work presented in [149] used average ridge clarity, number of manually annotated minutiae, ridge connectivity, minutiae reliability, and finger position to define the quality of latent fingerprint. The authors used a semi-automated quality assessment algorithm and achieved 80% quality prediction accuracy. However, their use of manually annotated minutiae makes their quality assessment results fraught with subjectivity. The method presented in [29] used number of minutiae, ridge clarity, core and delta, and ridge flow features for automatic latent value determination. Although their value determination algorithm required no manual feature markups, it still relied on manually marked ROI for segmentation . In [117], the authors used ridge clarity and ridge quality features to assess the quality of latent fingerprints.

Their approach required manually annotated minutiae and manually marked ROI. Chugh et. al [43] used a crowdsourcing based framework and multidimensional scaling to identify and understand how fingerprint experts assigned values to fingerprint images. They trained a prediction model that automatically assigned quantitative values to query latent fingerprints.

In our work, we use local features consisting of Gabor features, orientation certainty level, local ridge clarity, ridge frequency, ridge thickness, ridge-to-valley thickness, and spatial coherence to assess the quality of latent fingerprints. Unlike most of the other approaches that rely on manually segmented ROIs in the quality estimation process, our approach performs latent fingerprint quality assessment in a fully automated way. In the first stage of our approach, we segment the latent fingerprint ROIs using deep learning as described in Section 3.2.1. The segmented ROIs are split into 32x32 patches and local features are computed from the patches to build feature vectors used to train a multi-class perceptron classifier as detailed in Section 3.2.3. The classification results are used to assess the quality of the latent fingerprint. Note that this work does not consider overlapped latent fingerprints.

3.1.2 Contributions

This chapter makes the following contributions:

1. Poses latent fingerprint image quality assessment as a classification problem and solves it by using a deep neural network built by stacking RBMs. The depth chosen for our network was the one that gave the best performance and was found via experimentation. The depth is optimal for the problem being solved since going deeper did not yield appreciable performance gains and took longer to converge.

2. Unlike previous approaches, this work provides a region-of-interest based latent quality assessment strategy that requires no human intervention in latent fingerprint quality determination. The segmentation of the latent fingerprint and its quality assessment are done with no manual intervention or feature markups.

3.2 Technical Approach

Our latent fingerprint quality assessment architecture has two main stages. In the first stage, we use deep learning to segment the latent fingerprint. This stage involves feature learning, feature extraction and classification of the fingerprint patches into fingerprint and non-fingerprint classes. The segmented latent fingerprint referred to as the regions-of-interest (ROIs) consists of patches classified as fingerprint. In the second stage, we use a multi-class perceptron classifier to classify the fingerprint patches into three bins labelled 1 (good), 2 (bad) and 3 (ugly). The quality of the latent fingerprint is indicated by the label of the bin that contains the greatest number of patches. Ties are broken optimistically as explained in section 3.2.3. The block diagram of our proposed approach is shown in Figure 3.1.

3.2.1 Segmentation using Deep Learning

Restricted Boltzmann Machines (RBMs) are the building blocks for the proposed deep learning model. RBM is a stochastic neural network in which the nodes form an undirected bipartite graph. With RBM, a *k*-dimensional input can be mapped to a *j*-dimensional or *m*-dimensional feature space, where $j < k < m$. RBM has no intra-layer connections and given the visible unit activations, the hidden unit activations are mutually independent. Also the visible unit activations

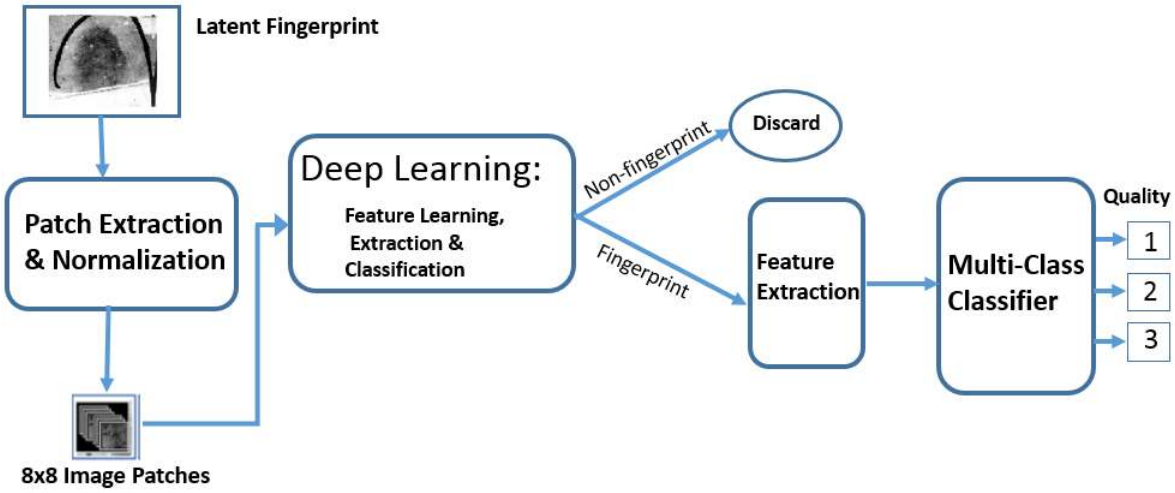


Figure 3.1: Proposed framework for latent fingerprint quality assessment. The first stage uses a deep learning architecture similar to that in [55], for feature learning, extraction and classification. In the second stage, features are extracted from the segmented fingerprint (ROI) and fed to a multi-class perceptron classifier. The target values from the classifier are 1 (Good), 2 (Bad) and 3 (Ugly).

are mutually independent given the hidden unit activations [35]. These characteristics of RBMs make them ideal for identity mapping. From experiments, neural networks built with RBMs are suitable for learning input representations that can be used to reconstruct the inputs with minimal reconstruction error. This makes such networks attractive for patch based latent fingerprint segmentation.

The segmentation stage of the proposed model is similar to that in [55]. In this stage, latent fingerprint image is partitioned into 8x8 non overlapping patches. Stochastic features that model a distribution over image patches are learnt using a generative multi-layer feature extractor. The features are used to train a single layer perceptron classifier that classifies the patches into fingerprint and non-fingerprint classes. The fingerprint patches are used to reconstruct the latent fingerprint image and the non-fingerprint patches which contain the structured noise in the original latent fingerprint are discarded. The segmented latent fingerprints from this stage are used as inputs to the quality assessment stage. The choice of patch size of 8x8 for the segmentation stage is based

on its optimality [53].

3.2.2 Network Hyper-Parameters

The values of the parameters used in the proposed segmentation and quality assessment networks are shown in Table 3.1 and Table 3.2, respectively. The values were selected through experiments.

3.2.3 Quality Assessment

In the quality assessment stage, 32x32 patches are extracted from the segmented (only fingerprint segments) ROIs and features computed from them are used as the quality assessment training dataset. The choice of 32x32 is based on the fact that for 500 pixels per inch (ppi) images, the width of a pair of ridge and valley is 8 to 12 pixels wide [97]. This implies that a patch size of at least 24x24 pixels is required to cover two ridges with a valley in between.

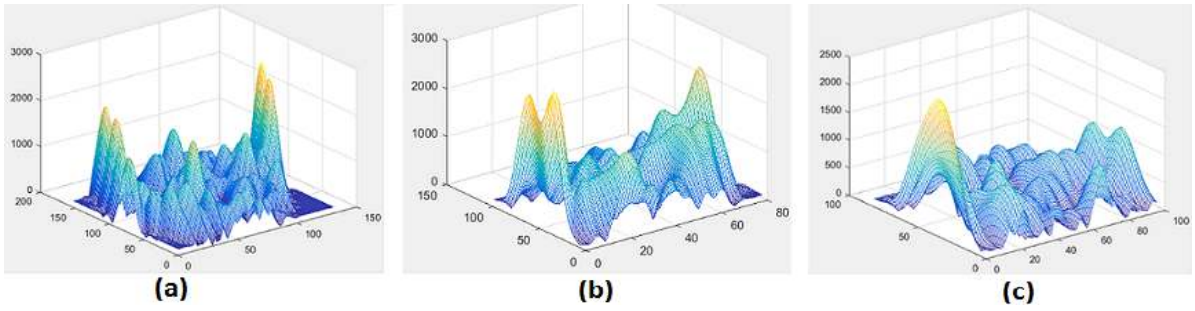


Figure 3.2: Gabor magnitude responses to sample segmented fingerprints : (a) Good (b) Bad, and (c) Ugly). As can be seen from the figures, good quality patches have more well-defined peaks than the bad and ugly patches. Also the peaks in (b) are more distinctive than in (c).

Given a segmented latent fingerprint image *L*, let *g, b, u* be the number of its 32x32 patches classified into bins B_1, B_2, B_3, respectively. Let $val = max\{g, b, u\}$. The quality of *L* is

Segmentation Network							
Parameter	L_1	L_2	L_3	L_4	L_5	L_6	L_7
Number of Neurons	64	800	1000	1200	1024	1024	2
Batch Size	-	100	100	100	100	100	-
Epochs	-	50	50	50	50	-	-
Learning Rate	-	1e-3	5e-4	5e-4	5e-4	5e-4	-
Momentum	-	0.70	0.70	0.70	0.70	0.70	-
Iteration	-	-	-	-	-	50	-

Table 3.1: Parameters and values for segmentation network. L_i refers to layer *i*. L_1 is the input layer. Layers 2, 3, 4 and 5 are RBM layers. L_6 is the perceptron layer and L_7 is the output layer

Quality Assessment Network			
Parameter	$Input\ Layer$	$Hidden\ Layer$	$Output\ Layer$
Number of Neurons	64	450	3
Batch Size	-	32	-
Epochs	-	10	-
Transfer function	-	logsig	tansig

Table 3.2: Parameters and values for the quality assessment network.

defined as:

$$Q(L) = \begin{cases} 1, & \text{if val = g;} \\ 2, & \text{if val = b;} \\ 3, & \text{if val = u.} \end{cases} \tag{3.1}$$

Ties are broken in an optimistic manner. For example, if $g = b$ and $b > u$, then $Q(L) = 1$.

Features used for Quality Assessment

The local features used for latent fingerprint quality estimation are shown in Table 3.3.

Features	**Description**
Peak Kurtosis Mean Kurtosis	Kurtosis of image patch magnitude and phase response
Peak Skewness Mean Skewness	Skewness of image patch magnitude and phase response
Ridge frequency [92] Ridge thickness Ridge-to-valley thickness	Values computed from a sinusoidal model of ridges and valleys in the image patch
Orientation certainty level	Measure of orientation strength
Spatial coherence	Computed from the gradient of image patch

Table 3.3: Local features used for latent quality assessment.

We use kurtosis and skewness of the magnitude and phase of Gabor filter response to measure the local quality of an image patch. Skewness is defined as a measure of symmetry [1]. A distribution is symmetric if the left and right sides of its central point are similar. Kurtosis is

a normalized form of the fourth central moment of a distribution and a measure of how heavy-tailed or light-tailed a given distribution is relative to a normal distribution. Given a vector $V = \{v_1, v_2, \cdots, v_k\}$, the skewness *S* and kurtosis *K* are defined as:

$$S = \frac{\frac{1}{|V|}\sum_{j=1}^{|V|}(v_i - \hat{v})^3}{\sigma_3}, \tag{3.2}$$

$$K = \frac{\frac{1}{|V|}\sum_{j=1}^{|V|}(v_i - \hat{v})^4}{\sigma_4}, \tag{3.3}$$

where σ, and $\hat{v}$ are the standard deviation, and mean, respectively. From experiments, we found that the areas of a fingerprint image with a regular ridge-valley patterns tend to have a high Gabor filter magnitude responses while those with unclear ridge-valley patterns have low and sometimes constant Gabor magnitude filter responses. Figure 3.2 shows the Gabor filter magnitude responses for sample good, bad and ugly segmented latent fingerprints from our deep learning model and shows the discriminative potential of the selected Gabor features for classifying patches into good, bad and ugly bins.

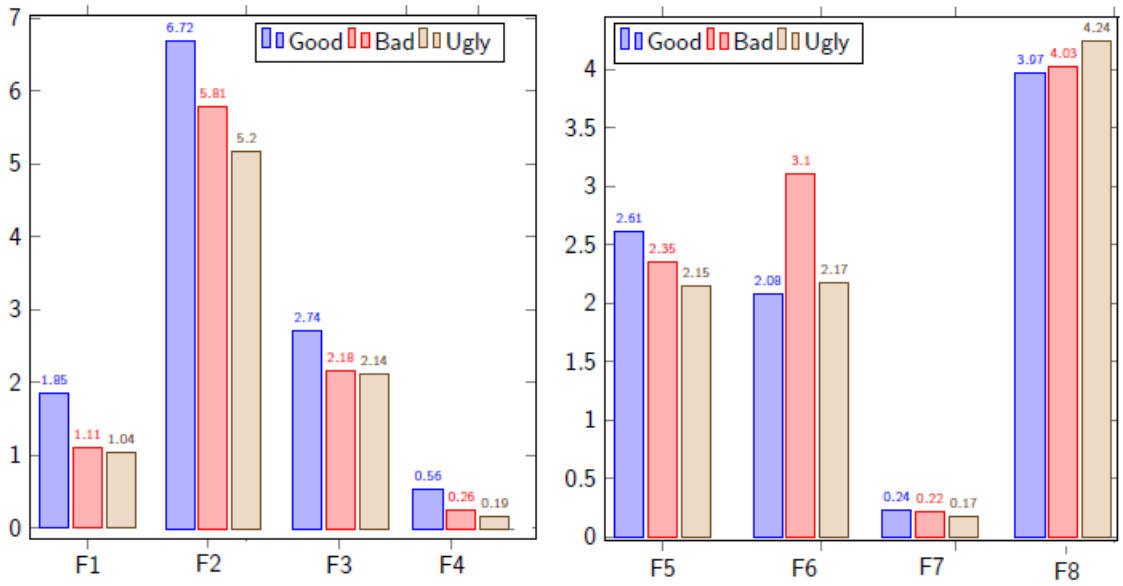


Figure 3.3: Gabor features used to train the multi-class perceptron classifier for image patch quality assessment. The features were computed from the kurtosis and skewness of the Gabor filter responses of the image patches. F1 and F4 are the peak and mean skewness of the magnitude response, respectively. F2 and F5 are the peak and mean kurtosis of the phase response, respectively. F3 is the mean kurtosis of the magnitude response. F6 and F7 are the peak and mean skewness of phase response, and F8 is the peak of the magnitude response. F7 was scaled up by 0.2 for visibility. The charts show that together, the features exhibit discriminative potential for classifying patches into good, bad and ugly bins.

3.3 Experiments and Results

We implemented our algorithms in Matlab R2014a running on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. Our implementation relied on NNBox (a Matlab toolbox for neural networks, and multi-class perceptron with Levenberg-Marquardt optimization. We evaluated our model on NIST SD27 [7] latent fingerprint database. The 258 latent fingerprint images in NIST SD27 consists of 88 Good, 85 Bad and 85 Ugly quality latent fingerprint images. The quality assigned to each image was based on the condition of the image in the location in which the minutia was positioned and on how clearly identifiable the type of the minutia was in the image [7]. The

results of the segmentation and quality assessment stages of our network were evaluated using the NIST SD27 ground-truth quality datasets from [53]. We generated the ground-truth dataset used in evaluating the results of the quality assessment stage. The details are provided in section 3.3.2.

3.3.1 Performance Evaluation Metrics

We used the following metrics to evaluate the performance the segmentation and quality assessment stages of our network.

- **Missed Detection Rate (*MDR*):** This is the percentage of class $\mathcal{C}_1$ patches classified as class $\mathcal{C}_2$ patches and is defined as.

$$MDR = \frac{FN}{TP + FN} \tag{3.4}$$

  where *FN* is the number of false negatives and *TP* is the number of true positives.

- **False Detection Rate (*FDR*):** This is the percentage of class $\mathcal{C}_2$ patches classified as class $\mathcal{C}_1$ patches. It is defined as:

$$FDR = \frac{FP}{TN + FP} \tag{3.5}$$

  where *FP* is the number of false positives and *TN* is the number of true negatives.

- **Segmentation Accuracy (SA):** It gives a good indication of the segmentation reliability.

$$SA = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.6}$$

  For the segmentation stage, $\mathcal{C}_1$ = fingerprint, $\mathcal{C}_2$ = non-fingerprint, and for the quality assessment stage, $\mathcal{C}_1 \in \{\text{Good, Bad, Ugly}\}$ and $\mathcal{C}_2 \in \{\text{Good, Bad, Ugly}\}$. We also used **precision** and **recall** to evaluate the performance of classifier used for quality assessment.

- **Precision:** Precision is the percentage of examples that truly belong to class k among all examples that the classifier predicted as belonging to class k.

- **Recall:** Recall is the percentage of examples correctly predicted as belonging to class k among all examples that truly belong to class k.

3.3.2 Latent Fingerprint Database

The ROI segmentation and quality assessment stages of our model were trained, validated and tested on NIST SD27 latent fingerprint databases. This database contains images of **258** latent crime scene fingerprints and their matching rolled tenprints. The images are grouped into good, bad or ugly categories. The grouping is based on the quality of the image determined by latent examiners. NIST SD27 has **88** Good, **85** Bad and **85** ugly quality latents. The latent prints and rolled prints are at **500** ppi.

Segmentation: Training, Validation and Testing

The training, validation and testing of the segmentation part of the model was done with 232,000 8x8 patches (132,000 for training, 50,000 for validation and 50,000 for testing) from the NIST SD27 database with 40% from good 30% from bad, and 30% from ugly NIST image categories. Table 3.4 shows the confusion matrix reflecting the results of training, validation and testing. We did not notice any appreciable performance gain when the model was trained with more than 132,000 patches.

		Predicted Patch Class (Training)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**23,665**	**11**
	Non-Fingerprint	**0**	**108,324**

		Predicted Patch Class (Validation)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**13,637**	**163**
	Non-Fingerprint	**2**	**36,198**

		Predicted Patch Class (Testing)	
		Fingerprint	Non-Fingerprint
Actual Patch Class	Fingerprint	**13,914**	**188**
	Non-Fingerprint	**5**	**35,893**

Table 3.4: NIST SD27 - Confusion matrix for training, validation and testing for the segmentation stage.

Figure 3.4 shows the segmentation results of our proposed method on sample good, bad and ugly quality images from the NIST SD27 database. It shows the original latent fingerprint images and the segmented fingerprints constructed using patches classified as fingerprints.

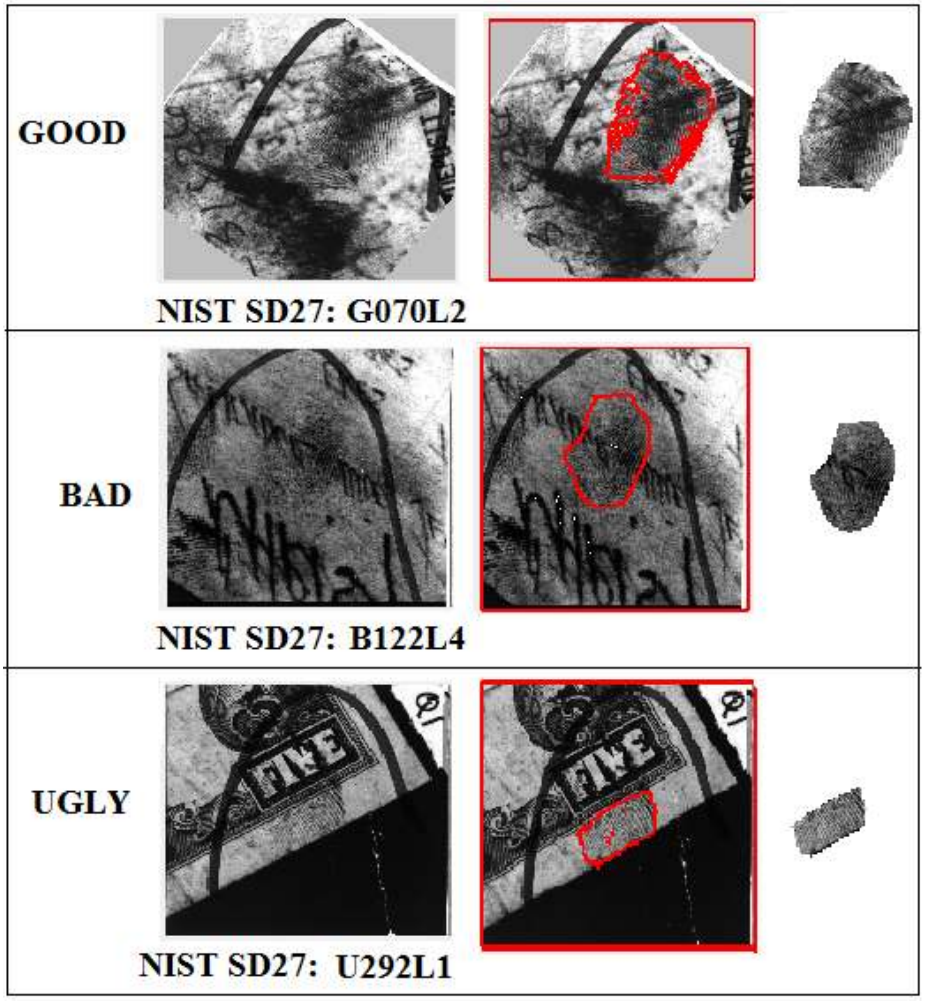


Figure 3.4: NIST SD27: Segmentation results without post classification processing for Good (row 1), Bad (row 2) and Ugly (row 3) latents. Each row shows the original image followed by an outline of the segmented fingerprint superimposed on the original image, and the segmented fingerprint only part. The segmented fingerprint part was constructed with patches classified as fingerprint.

Stability of the Segmentation Network

The stability of the segmentation network was investigated by selecting 40 images at random from each (Good, Bad and Ugly) category of NIST SD27 database and extracting 50,000 8x8 patches from each category for a total of 150,000 8x8 patches. We performed 5 runs of network training and for each of the 5 runs, we used 20,000 patches randomly sampled from the 150,000

patches. All the model parameters (number of epochs, number of iterations etc.) shown in Table 3.1 remained unchanged across the runs. The mean square reconstruction error (msre) and mean error cost at convergence, as well as the standard deviation for the 5 runs are shown in Table 3.6. Plots of the error during training for each run are shown in Figure 3.5. These results indicate that the proposed segmentation model is stable.

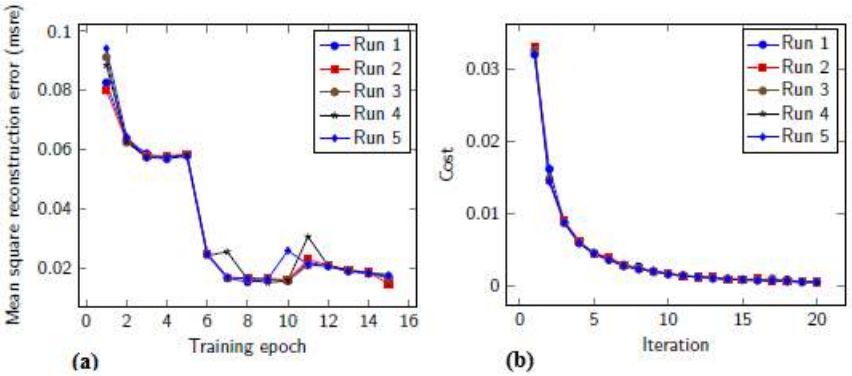


Figure 3.5: Segmentation Network Stability: (a) shows that the mean square reconstruction error (MSRE) during the pre-training phase for the 5 runs followed similar trajectories. Similarly, (b) shows that the error during the fine-tuning phase for the 5 runs were close. These results are indicative of the stability of the network.

Run #	MSRE	Cost	MDR	FDR
1	0.0169	5.769E-04	3.220E-04	1.10E-05
2	0.0159	5.406E-04	3.950E-04	0.00
3	0.0148	5.420E-04	1.720E-04	0.00
4	0.0167	5.562E-04	3.310E-04	1.20E-05
5	0.0175	5.145E-04	2.091E-04	0.00
Mean	0.01636	0.00055	2.8E-04	4.6E-06
Standard Deviation	0.00104	2.289E-05	9.235E-05	6.309E-06

Table 3.5: Segmentation Network Stability:: The mean square reconstruction error (MSRE) at convergence during the pre-training phase, cost at convergence during the fine-tuning phase, MDR, and FDR for the 5 different runs are close. The mean and standard deviation indicate stability across the 5 runs.

Quality Assessment: Training, Validation and Testing

There are 258 latent fingerprint images in NIST SD27 database with 88, 85 and 85 in the Good, Bad and Ugly categories [74]. The 258 latent fingerprint images were segmented using our trained segmentation model. We extracted 7,000 32x32 patches from 50 Good, 50 Bad and 50 Ugly ROIs for training, 1,500 32x32 patches from 20 Good, 20 Bad and 20 Ugly ROIs for validation, and 1,500 32x32 patches from 18 Good, 15 Bad and 15 Ugly ROIs for testing. The multi-class perceptron classifier (MPC) in the quality assessment stage of our model was trained, validated and tested with the training, validation and testing datasets that were independently drawn from the fingerprint only segments from the 258 latent fingerprint images in NIST SD27 database.

To label the patches in the 32x32 patch datasets, we computed the average fractal dimension (FD_{av}) and fractal dimension spatial frequency (FD_{sf}) for each patch. The label L_p for each

patch p was determined using equation 3.7. The thresholds used in equation 3.7 were empirically determined. Figure 3.6 shows sample patches and their FD_{av} and FD_{sf}.

$$L_p = \begin{cases} 1, & \text{if } \tau > 1.75 \text{ and } \kappa < 0.65; \\ 2, & \text{if } 1.65 < \tau < 1.75 \text{ and } 0.65 < \kappa < 0.70; \\ 3, & \text{if } \tau < 1.70 \text{ and } \kappa > 0.70. \end{cases} \tag{3.7}$$

where τ and κ are the FD_{av} and FD_{sf} of patch p, respectively. Figure 3.7 shows the confusion matrix for MPC training, validation and testing, as well as the validation performance and error histogram on NIST SD27.

1	2	3	4	5	6	7	8	9	10	11
Image Patch										
FD_{av}	1.773	1.804	1.767	1.675	1.737	1.727	1.590	1.633	1.645	1.512
FD_{sf}	0.611	0.617	0.649	0.669	0.675	0.659	0.713	0.772	0.768	0.825

Figure 3.6: Image patches from NIST SD27 with their computed average fractal dimension (FD_{av}) and fractal dimension spatial frequency (FD_{sf}). Patches with visible fingerprint patterns (columns 2, 3, & 4) have higher average FD and lower FDsf, than those with little visible fingerprint patterns(columns 5, 6 & 7) or no visible fingerprint patterns (columns 8, 9, 10 & 11). The higher the FD_{av} and the lower the FD_{sf}, the better the quality of the patch, and conversely.

The quality assessment model achieved a quality assessment accuracy of 96.1% for Good, 91.1% for Bad and 96.7% for ugly latent fingerprints on the testing dataset, and 97.9% for Good, 92.7% for Bad and 97.5% for ugly latent fingerprints on the validation dataset.

Stability of the Quality Assessment Network

To investigate the stability of the quality assessment network, we performed 5 runs of training, validation and testing of the network using the dataset created in 3.3.2. All the model

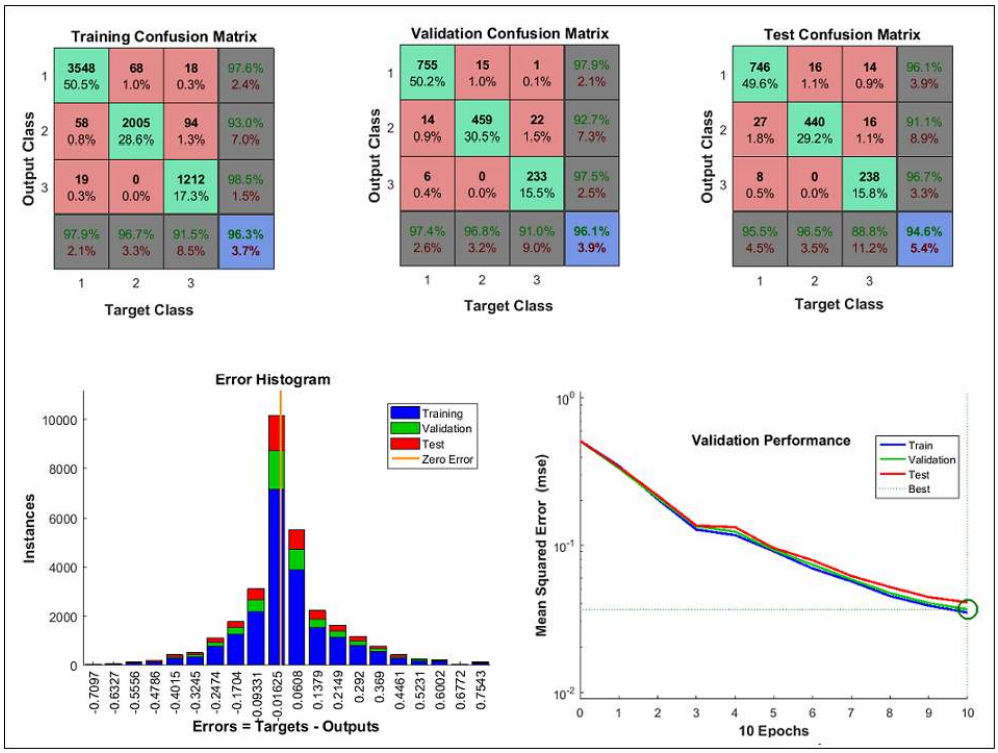


Figure 3.7: NIST SD27 - Confusion matrix for training, validation and testing, error histogram, and validation performance for the quality assessment neural network. Class 1 = Good, Class 2 = Bad, and Class 3 = Ugly. 7,000 patches were used for training, 1,500 patches for validation and 1,500 patches for testing. The training, validation and testing samples were independently drawn from the dataset. Output Class is the predicted class while target class is the ground-truth class. The fourth row contains Recall values while the fourth column contain the Precision. In the testing confusion matrix, Precision=96.1% and Recall=95.5% for Class 1 means that out of the times Class 1 was predicted, the classifier was correct 96.1% of the time, and out of all the times Class 1 should have been predicted 95.5% of the predictions were correct. The small numbers on all cells but the diagonal (that contains the true positives for the respective classes), as well as the error histogram, and validation performance plots, indicate good classifier performance.

parameters (number of epochs, number of iterations etc.) remained unchanged across the runs. The overall precision for training, validation and testing, as well as the mean accuracy, and standard deviation for the five runs are shown in Table 3.6. The precision and recall for the three classes in the five runs are provided in Table 3.7. These results indicate the stability and reliability of the network.

Run #	Training	Validation	Testing
1	93.9%	93.1%	93.6%
2	96.3%	96.1%	94.6%
3	95.5%	94.0%	94.0%
4	97.6%	96.4%	95.8%
5	94.9%	94.1%	93.8%
Mean (%)	95.46	94.74	94.36
Standard Deviation (%)	1.12	1.44	0.89

Table 3.6: Network Stability: The precision values (computed with the true positives along the diagonal of the confusion matrix) in each column are close. The mean and standard deviation indicate stability across the five runs.

Validation: Run #	Class	Precision	Recall	Testing: Run #	Class	Precision	Recall
1	1	91.9%	97.3%	1	1	93.0%	98.0%
	2	93.8%	88.1%		2	93.2%	89.6%
	3	95.7%	89.7%		3	96.0%	87.3%
2	1	97.9%	97.4%	2	1	96.1%	95.5%
	2	92.7%	96.8%		2	91.1%	96.5%
	3	97.5%	91%		3	96.7%	88.8%
3	1	92.1%	99.2%	3	1	92.1%	99.1%
	2	94.4%	88.7%		2	94.2%	88.0%
	3	100%	88.7%		3	100%	88.9%
4	1	98.7%	96.7%	4	1	98.4%	97.6%
	2	92%	98.3%		2	93.7%	98.8%
	3	97.1%	92.8%		3	97.0%	92.4%
5	1	96.8%	95.7%	5	1	96.7%	96.5%
	2	90.6%	95.3%		2	88.2%	95.3%
	3	95.2%	92.8%		3	95.2%	85.4%

Table 3.7: Network Stability: Precision and Recall for the three classes in the five runs during validation and testing. For both the validation and testing samples, there is no marked difference between the Precision for a given class from one run to the next. The same applies to the Recall. This indicates that the network is reliable.

3.3.3 Evaluation of Latent Quality Prediction

We compare the latent quality predictions of the proposed model with the VID, VEO, and NV value determination by latent examiners [70] as well as the quality value predictions by Expert Crowd [43]. Note that NIST SD27 database is the only latent database with available latent

value determinations by latent examiners. As reported in [70], there are 210 VID, 41 VEO , and 7 NV latents in NIST SD27. A total of 166 latents (155 VID , and 11 VEO) out of the 256 latents in NIST SD27 are retrieved at Rank-1 using state-of-the-art latent AFIS [43]. To ensure a fair comparison between the three results being compared, we follow the protocol used in [43]. The 258 latents are sorted in ascending order of the quality [1-3] predicted by our model, and then partitioned into three, P_1, P_2, P_3. Partition P_1 contains the first 210 latents, P_2 contains the next 41, while P_3 contains the last 7 latents. Following [70], the latents in P_1, P_2 and P_3 are considered as VID, VEO and NV, respectively. Table 3.8 shows a comparison of the number of latents retrieved at rank-1 using value determination by latent examiners [70], the predicted latent value from [43], and the predicted latent quality from our quality prediction model. A reference dataset containing 2,257 rolled prints created from 2,000 fingerprint images in NIST SD4 database, and the 257 rolled images in NIST SD27 database was used for this performance comparison. In terms of predicting latent AFIS performance, the quality prediction by our model is better than the value determination latent examiners and value prediction by Expert Crowd. 164 latents predicted by our model as VID latents were identified at Rank-1 compared to 161 identified at Rank-1 based on value determination by Expert Crowd.

	VID	**VEO**	**NV**
Latent Examiners [70]	155/210	**11**/41	0/7
Expert Crowd [43]	161/210	5/41	0/7
This Work	**164**/210	4/41	0/7

Table 3.8: NIST SD27 latent fingerprints retrieved at Rank-1 using a state-of-the-art latent AFIS. The results show that the proposed quality assessment model performs slightly better than Expert Crowd [43] in predicting latent AFIS performance for VID latents (164 vs 161). However, both Latent examiners and Expert Crowd are better than the proposed model in predicting latent AFIS performance for VEO latents.

3.4 Summary

We proposed automatic region-of-interest based latent fingerprint quality assessment technique using deep learning. The first stage of our proposed method uses feature learning, extraction and classification to segment the latent fingerprint image. In the second stage, 32x32 patches are extracted from the segmented ROI image and features computed from them are fed to a multi-class perceptron that classifies each fingerprint patch into Good, Bad or Ugly bins. The quality of a latent fingerprint is indicated by the label of the bin that contains the greatest number of patches, with ties broken optimistically (if the number of patches in the Good bin is equal to that in the Bad bin and greater the number in the Ugly bin, the quality of the fingerprint is set to Good). We demonstrated the performance of our model on the NIST SD27 latent fingerprint database. We presented a comparative analysis showing that in terms of predicting latent AFIS performance, the quality prediction by our model performs better than the state-of-the-art latent fingerprint value determination model.

Chapter 4

DeepLatent: A Deep Learning Model for Patch Based Latent Fingerprint Matching

Fingerprints have been one of the most reliable methods used in forensics for human recognition. Latent fingerprints are usually partial fingerprints and are characterized by few minutiae points, and missing singular points such as core and delta. The dearth of those structures coupled with unspecified orientations, distortions, variations in the illumination of a crime scene, and occlusions, make matching latent fingerprints to full rolled/plain fingerprints very challenging. Many of the existing approaches for matching latent fingerprints to rolled/plain fingerprints [76, 59, 75, 106, 99, 49, 18] rely on fingerprint features mentioned above and become unreliable when the latent fingerprint does not include those structures. In [76, 59, 75], the authors use minutiae or combination of pores, and ridge features and minutiae. A descriptor-based Hough transform

method was used in [106]. In [99], the authors use local minutiae clustering with multiple alignments. Matching based on score level fusion using min, max, sum, and product of minutiae, quality map, and orientation was used in [49]. An automated feedback mechanism was used in [18] to refine the set of features that are similar between the probe and reference fingerprints. Unlike the above methods, our approach computes similarity scores of probe and reference image patches by taking into account the overlapped areas on the latent and rolled fingerprints, and then matching the minutiae on them, if available. The overall idea of this chapter is captured in Figure 4.1.

Patch-based image matching has been extensively used in computer vision tasks. It has been used for finding accurate correlation between images in domains such as object recognition [95], classification [147], image stitching [28], and image reconstruction [122]. Our approach to patch similarity learning is similar to the techniques used in [76], [65] and [152]. The main difference is that before using a neural network to learn the pairwise similarity between image patches, we first transform the patches into a frequency domain representation. To the best of our knowl-

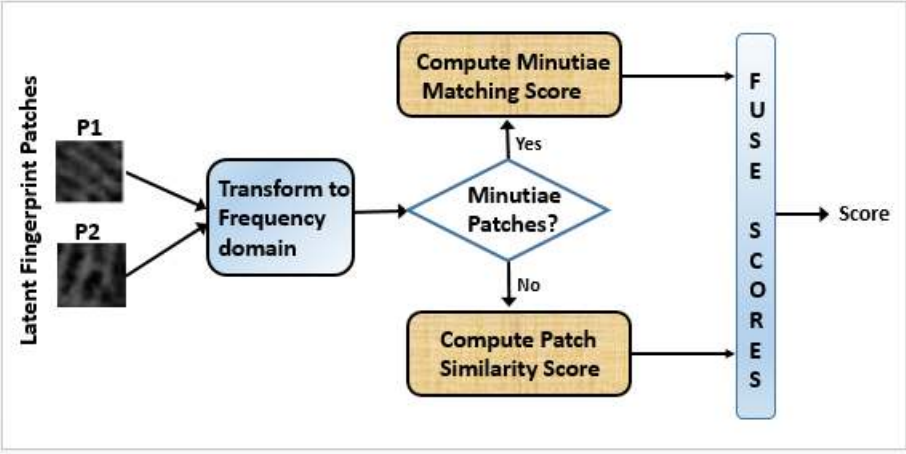


Figure 4.1: Latent fingerprint patch pair (P1,P2) are first transformed into a frequency domain representation. We compute minutiae matching score if P1 and P2 are minutiae patches, or patch similarity score, otherwise. The scores for all patch pairs are fused to obtain the matching score.

edge, this work represents the first attempt at performing latent fingerprint matching by learning similarity of image patches and minutiae in correlated patches from frequency domain representation of image patches using convolutional neural networks. Unlike the previous work, we learn a similarity function directly from frequency domain representation of annotated pairs of raw image patches using a deep neural network modeled after a Siamese network (a neural network architecture consisting of two or more identical networks). The use of frequency domain representation is informed by the fact that periodicity and relative strengths of periodic components in input fingerprint data are readily revealed in frequency domain. Those frequency characteristics are pertinent to differentiating or correlating fingerprint image patches. The similarity learning is done using the Euclidean distance as a measure of similarity. A descriptor is learnt using patch pairs with the objective of minimizing the L_2 norm between similar patches and maximizing the same between two non-similar patches. Multiple experiments are conducted using both spatial and frequency domain representations of patches. The results obtained show that using frequency domain representations of patches results in significant improvement in the accuracy of of the proposed model. We also explored different architectures for DeepLatent by evaluating their performance on a subset of the training and validation dataset.

The rest of the chapter is organized as follows: Section 4.1.1 presents a review of recent works in latent fingerprint matching while section 4.1.2 describes the contributions of this chapter. Section 4.2 highlights our technical approach and describes the networks that constitute DeepLatent. It also highlights frequency domain representation. Section 4.2.2 provides an overview of patch based matching. The experimental results and performance evaluation of our proposed approach are presented in Section 4.3, while Section 4.4 contains the conclusions and future work.

4.1 Related Work and Contributions

4.1.1 Related Work

Many of the existing approaches for matching latent fingerprints rely on features extracted from the latent fingerprints. Jain et al. [75] proposed a latent-to-rolled/plain matching algorithm that relied on manually marked features (minutiae, core, delta) for the latent fingerprints in NIST SD27 database and automatically extracted features for rolled prints in NIST SD4 and NIST SD14 databases. They reported a rank-1 identification rate of 74 percent. Feng et al. [59] used ridge pattern, singular points, and orientation field to match latent fingerprints in NIST SD27 with a database of 10,258 rolled fingerprints. They reported a rank-1 accuracy of 73.3%. Tsai et. al [76] used localized secondary features derived from relative minutiae information and trained a neural network to generate the final similarity score based on minutiae matched in the overlapping areas of a query latent fingerprint and reference fingerprints. They reported 1.21% and 0.68% improvements on minimum total error rates of FVC2002 DB1 and DB2 databases, respectively. Deep learning has successfully been applied to latent fingerprint image segmentation [55], enhancement [90], and matching [56].

Previous works have used frequency representation to train deep learning models. Zou et al. [158] used restricted number of frequency coefficients of Discrete Cosine Transform (DCT) of images of handwritten digits to train a deep belief network for handwritten digit recognition. Er et al. [51] trained a Radial Basis Function network for face recognition using DCT features computed from face data. Ulicny et al. [136] used DCT transformation of raw images to train CNN to classify images encoded in compressed form. In our approach, image patches are transformed to frequency

Author	Approach	Databases	Database Size	Rank-1 (%)	Rank-20 (%)
Jain et al. [75]	Singularity, ridge quality map, Rank-k matching, ridge flow map, ridge wavelength map, and skeleton	NIST SD4, NIST SD14, NIST SD27	29,257	74.0	82.90
Paulino et al. [106]	Descriptor-based Hough transform, manually marked minutiae, orientation field, similarity	NIST SD4, NIST SD27	2,258	62.40	-
Feng et al. [60]*	Fusing the same features from different samples of rolled fingerprints and plain fingerprints	-	4,180 pairs	57.80 (plain) 70.40 (rolled) 83.0 (boosted max. fusion)	-
Dvornychenko [49]	Fusing results from five different latent fingerprint matchers	-	-	-	-
Arora et al. [18]	Ridge orientation and frequency, latent feature refinement using information in exemplars (feedback)	NIST SD14, MSP DB	100,000	49	59.5
Medina-Pérez et al. [99]	Local minutiae clustering using multiple alignments	NIST SD4, NIST SD14	29,000	68.60	-
This chapter	Image patch similarity, and matching minutiae on correlated patches	NIST SD4, NIST SD27, Synthetic DB	29,257	*78.56*	*87.45*

Table 4.1: Recent work in latent fingerprint matching showing the various approaches that have been used. MSP stands for Michigan State Police. * Matched 230 latents in the ELFT-EFS Public Challenge Dataset against a database of 4,180 pairs of rolled and plain fingerprints.

domain using Fast Fourier Transform (FFT) and the transformed data are used to train CNNs for patch similarity learning, minutiae detection and matching.

4.1.2 Contributions

Researchers have used different approaches for latent fingerprint matching. Most of the strategies are captured in a survey of latent fingerprint matching [118]. Table 4.1 shows recent works in latent fingerprint matching and the various approaches. As can be seen from Table 4.1, most of the previous efforts in latent fingerprint matching achieved moderate accuracy, hence the need for

new a algorithm for improved latent fingerprint matching. The contributions of this chapter are:

- A novel system for patch-based latent fingerprint matching using deep neural networks with an improvement on the previous latent fingerprint matching results. The proposed system is a unified framework for patch based similarity learning, minutiae detection and matching, without relying on hand-crafted features.

- A novel hierarchical matching algorithm that uses single-step matching or two-step matching based on the quality of the probe latent fingerprint image.

- Rigorous experiments that demonstrate the superiority of using frequency domain representation of fingerprint image patches for similarity learning and minutiae detection.

This chapter extends our preliminary work [56], but differs significantly from it in the following aspects: (a) A hierarchical patch based matching algorithm for single-step or two-step matching based on the quality of the latent fingerprint image. (b) A unified frequency domain based framework for minutiae detection, matching and patch similarity learning. (c) A Siamese hybrid network with a neighborhood relationship computation and summarization layer, for determining how similar two minutiae patches are based on the relationship between the pixels in the neighborhood of the minutiae. (d) In-depth theoretical and empirical discussions on the performance the proposed model when trained with frequency domain representation versus spatial domain representation of fingerprint image patches. (e) Detailed investigation of the best combination of features (correlation, minutiae, relations and frequency) that yields the best matching performance. (f) Empirical demonstration of the robustness of the proposed model to image rotation. (g) Evaluation of architectural variations to determine the impact of feature dimensionality on the accuracy and

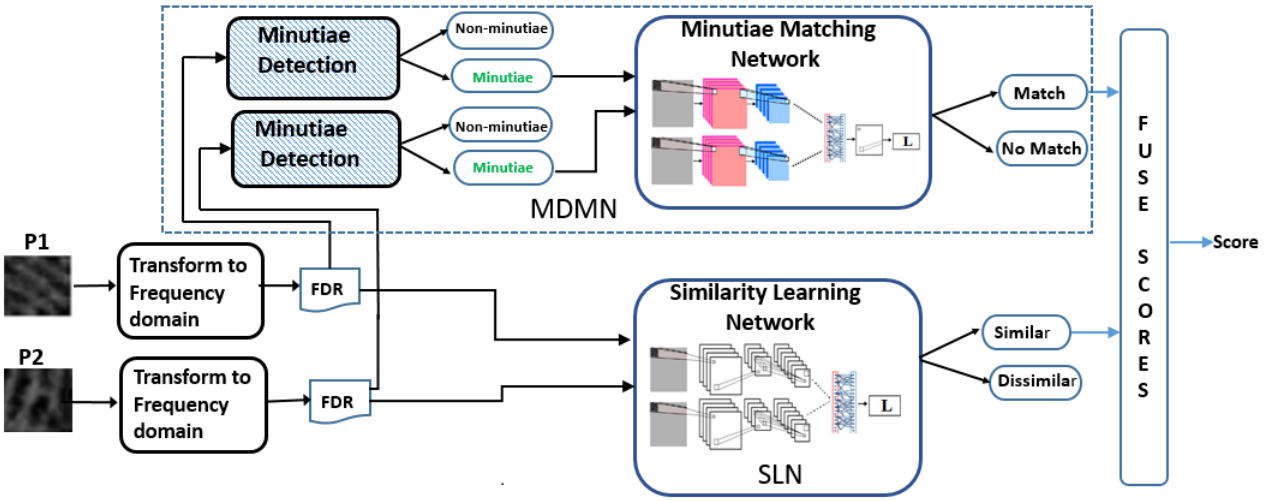


Figure 4.2: Proposed architecture for DeepLatent consisting of minutiae detection and matching network (MDMN) and patch similarity learning network (SLN). MDMN is used for detecting minutiae in patch representations, and determining if two minutiae patches match. SLN is for learning the similarity between representations of two patches. Section 4.2.1 and section 4.2.1 provide more details on the architectures of the two networks.

computational burden of the proposed framework. (h) Evaluation of the matching performance of the proposed method performed on a larger reference database.

4.2 Technical Approach

The block diagram of our proposed approach shown in Figure 4.2 consists of two convolutional neural network (CNN) models. The first model is the minutiae detection and matching network (MDMN) used for detecting minutiae in patch representations, and determining if two minutiae patches match. The second model is a similarity learning network (SLN) for learning the similarity between the representations of two patches. SLN learns a distance metric for determining the correlation between the representations of two patches. During training, a pair of MDMN similar to the Siamese network, is applied to the frequency domain representations of pairs of patches

(P1, P2). The MDMN detects the presence or absence of minutiae in the patches using the representations and computes the matching score for pairs of minutiae patches. The same representations of the patches are fed to a Siamese similarity learning network (SLN). Both networks are trained on the patch-pairs generated from good quality images from the NIST SD4 database. Table 4.2 shows the hyper-parameters and values for the MDMN. The structure of the SLN is depicted in Table 4.3. The parameters for both MDMN and SLN were selected based on the performance of the network on their respective validation sets. The choice of 32x32 patch size is based on its empirically determined optimality. We use different networks for patch similarity learning and minutiae detection/matching because minutiae detection/matching needs a deeper and slightly more complex network for efficient minutiae detection and matching as well as learning the relationship between the structures around the minutiae.

The training and evaluation of the SLN and MDMN networks were done using $\approx$ 1.64M image patches (819,200 for SLN and 819,200 for MDMN). A dataset consisting of pairs of matching and non-matching patches were used for training and evaluating the SLN, while a dataset of pairs of matching and non-matching minutiae patches were used for the MDMN.

4.2.1 Proposed DeepLatent Networks

Minutiae Detection and Matching Network (MDMN)

Our minutiae matching network consists of two CNNs with shared weights. The architecture is similar to person re-identification CNN architecture in [12], but unlike [12], each branch of the MDMN has four layers of convolution and max pooling to learn a set of features used for detecting minutiae patches and comparing two minutiae patches. To capture information about the pixels

around a minutiae (which we refer to as relation attribute) that is very useful during matching, we compute neighborhood difference of the feature maps of patch pair (P_1, P_2) as $\Re = g(P_1) \bigoplus g(P_2)$ and $\hat{\Re} = g(P_2) \bigoplus g(P_1)$, where $\bigoplus$ is neighborhood difference operator. The computed neighborhood differences is summarized using a convolutional layer and the spatial relationship across neighborhood difference feature maps is learnt using 3x3 filters with stride 1 [12]. This is followed by a fully connected layer and finally a two output softmax layer. We use 3x3 filters to ensure that we capture local features essential for differentiating or correlating minutiae patches.

Name	**Type**	**Dim.**	**Filter**	**Stride**
Input	input	32x32x1	-	
Conv1	Convolution		8 3x3x1	
batchNorm1	Batch Normalization	8 channels		
ReLU1	ReLU			
MaxPool	Max Pooling	1x1		[1 1]
Conv2	Convolution		32 3x3x8	
batchNorm2	Batch Normalization	32 channels		
ReLU2	ReLU			
Conv3	Convolution		64 3x3x32	
batchNorm3	Batch Normalization	64 channels		
FC	Fully Connected Layer (2)			
Softmax	Softmax			
Output	Label output			

Table 4.2: Parameters for the MDCNN. Epochs: 20, Batch size: 64, Learning rate: 0.001.

Name	Type	Dim.	Filter	Stride
Input	input	32x32x1	-	
Conv	Convolution		30 5x5	[1 1]
ReLU	ReLU			
MaxPool	Max Pooling	1x1		[1 1]
FC	Fully Connected (2)			
Softmax	Softmax			
Output	Similarity output			

Table 4.3: Parameters for the similarity learning network (SLN). Epochs: 100, Batch size: 64, Learning rate: 0.01, Momentum: 0.9.

Patch Similarity Learning Network (SLN)

SLN is a 7-layer convolutional neural network consisting of a convolutional layer with 30 5x5x1 filters, max pooling, RELU activation functions units, 2 fully connected layers, and a softmax output layer. It has a receptive field of 32x32. For the architecture of SLN, we explored shallow and deep architectures and selected the architecture that had the least computational burden and best performance among the ones considered. The details of the accuracy/computation time and feature dimension tradeoff experiments that guided the selection of the SLN architecture are presented in Section 4.3.4.

Loss Functions

Given two patches P_1 and P_2, SLN outputs a real number between 0 and 1 that indicates how similar P_1 and P_2 are. SLN maps the learned representations of P_1 and P_2 to a low dimensional feature space so that the loss between similar pairs is minimized while that between two dissimilar

pairs is maximized. To achieve this, we trained the weights of the network using a margin based loss function defined as:

$$Loss(P_1, P_2, l) = l\mathbb{D}^2 + (1-l)\{max(0, m^2 - \mathbb{D}^2)\}, \tag{4.1}$$

where $l = 1$ if (P_1, P_2) is a positive pair and 0, otherwise. m is the margin of separation and is greater than 0 for negative pairs. $\mathbb{D} = ||g(P_1) - g(P_2)||_2$ is the Euclidean Distance between feature vectors $g(P_1)$ and $g(P_2)$ of patches P_1 and P_2. The minutiae detection/matching network was trained by minimizing the following loss function:

$$L(P_3, P_4, l) = LogLossSoftMax(z(P), l), \tag{4.2}$$

where P_3 and P_4 are two minutiae patches, $l = 1$ if (P_3, P_4) is a positive pair and 0, otherwise. P is the concatenation of the feature vectors of P_3 and P_4, and $z(.)$ is a function that output the matching scores.

Frequency domain representation

In many situations, image processing tasks are best performed in a domain other than the spatial domain. For such tasks, the images are transformed into the target domain and the tasks are performed in the transformed domain. Discrete Fourier Transform (DFT) can reveal periodicity and relative strengths of periodic components in input fingerprint data. We use Fast Fourier Transform which is a faster method for computing DFT. The 2-D frequency domain representation $\{F_k\} = F_0, F_1, \cdots, F_{K-1}$ of *K* image patches $\{p_k\} = p_0, p_1, \cdots, p_{K-1}$, where each $p_k, k = 1, \ldots, K-1$ is of size MxN is defined as:

$$F_{u,v} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p_{m,n} \cdot e^{-2\pi i(\frac{um}{M} + \frac{vn}{N})}, \tag{4.3}$$

where *u* and *v* are spatial frequencies in m and n directions, respectively, Each $F_k, k = 1, \ldots, K-1$ is also of size MxN.

4.2.2 Patch Based Matching

We define patch based match M_p between a patch from probe latent fingerprint and a patch from gallery fingerprint image as:

$$M_p = f(Q, C, F, M, \varphi), \tag{4.4}$$

where $f(.)$ is a function, Q, C, F, M, φ are the quality, patch similarity, spatial frequency, minutiae and relation attribute, respectively. The quality (Q) is an attribute of the probe latent patch determined using image quality assessment tool. Patch similarity and minutiae scores are obtained using convolutional neural networks. Spatial frequency is computed as in [91].

$$F = R_f^2 + C_f^2, \tag{4.5}$$

where R_f and C_f are the row and column frequency, respectively.

$$R_f = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=1}^{N-1} [G(x, y) - G(x, y-1)]^2, \tag{4.6}$$

$$C_f = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=1}^{N-1} [G(x, y) - G(x-1, y)]^2, \tag{4.7}$$

where M x N is the dimension of the image, and $G(x, y)$ is the gray value at position (x, y) in the image. Relation φ, is a measure of spatial relationship between pixels around the minutiae and is obtained from the summarization layer of the minutiae detection and matching network (MDMN). Details regarding φ and its computation are provided in Section 4.2.2. In this chapter, we use NIST Fingerprint Image Quality (NFIQ) [130] to determine Q in form of NFIQ number. NFIQ numbers

range from 1 to 5 with 1 being the best and 5 as the worst (1 = excellent, 2 = very good, 3 = good, 4 = fair, 5 = poor). The NFIQ number reflects how positively or negatively an individual sample contributes to the performance of a fingerprint matcher [130]. NFIQ 2.0 is a newer version of NFIQ tool with a wider range (1 to 100), but is not suitable for latent fingerprints [5].

We propose a hierarchical patch based matching algorithm for single-step M_p^1, or two-step M_p^2 matching defined as:

$$M_p^1 = f_1(C, M), \tag{4.8}$$

$$M_p^2 = f_2(C, M, F, \varphi), \tag{4.9}$$

M_p^1 is used for matching probe latent fingerprint images with NFIQ numbers 1 or 2 (very good to excellent quality), while M_p^2 is used for latent fingerprint images with NFIQ numbers 3, 4 or 5 (good to poor quality). The NFIQ numbers were computed using NIST open source fingerprint quality assessment tool (NFIQ).

Patch Similarity and Matching

In this work, we consider two patches to be similar if there exist an in-plane rotation by *d* degrees that makes them identical. When a given patch *p* is rotated by $d \in \{0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o\}$, we obtain a set of patches $\kappa = \{p_1, p_2, \cdots, p_8\}$. Each pair of patches $(p_i, p_k) \in \kappa$ are considered similar. This definition of similarity allows us to learn patch representations that are invariant to rotations and enhances the odds of finding a match for a given latent fingerprint patch. Figure 4.3 shows in-plane rotations of a sample fingerprint with a 32x32 patch highlighted with a bounding box. The 32x32 patches in all the rotations are similar.

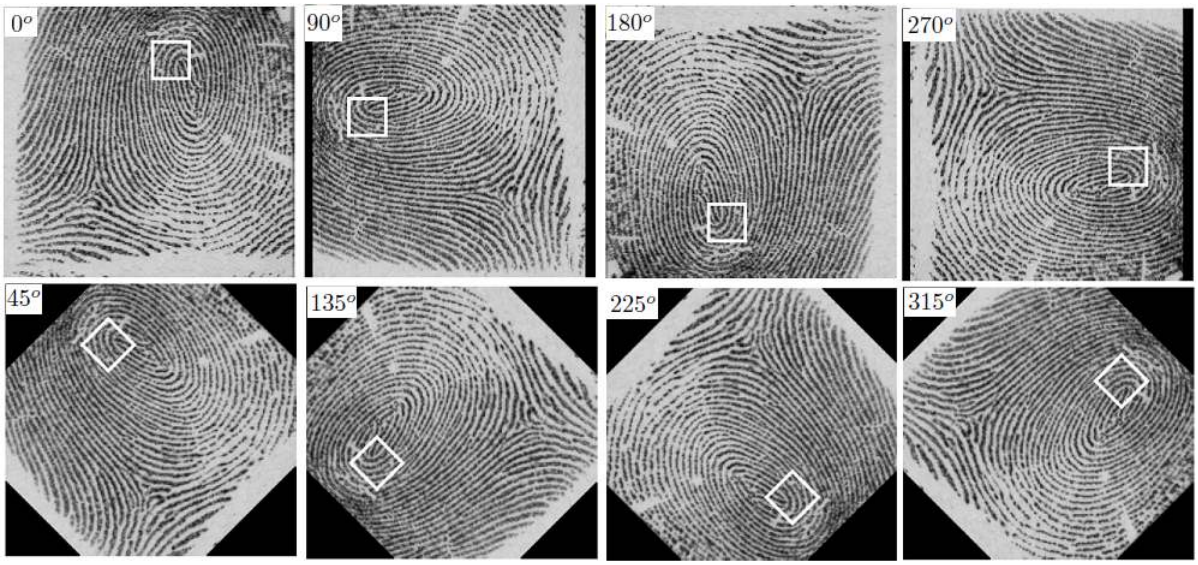


Figure 4.3: Various rotations $(0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o)$ of a sample fingerprint with a 32x32 patch highlighted with a bounding box. By our patch similarity definition, all the patches are similar.

Let *L* and *R* be probe and gallery fingerprints, respectively. Let $P_L = \{l_1, \cdots, l_k\}$ and $P_R = \{r_1, \cdots, r_n\}$ denote the 32x32 patches from *L* and *R*. For each $l_i \in P_L, i = 1, \ldots, k$, we create tuples $(l_i, r_j), r_j \in P_R, j = 1, \ldots, n$, feed each tuple to our trained model and record the similarity score. A reference fingerprint in the NIST SD27 database is 800x768 while one of the largest segmented fingerprint from a latent fingerprint in the same database is 380x448. The number of 32x32 non-overlapping patches from the reference and latent fingerprints are $\frac{614,400}{1,024} = 600$ and $\frac{170,240}{1,024} = 166$, respectively. Evaluating a match between them requires $166 \times 600 = 99,600$ comparisons in the worst case. To minimize the computation time, a pipeline with 166 parallel computations can be used. The patch similarity score for each $l \in P_L$ is defined as:

$$S_l = max(s(l, r_1), \ldots, s(l, r_n)) \tag{4.10}$$

where $s(l, r_k)$ is the similarity score of tuple $(l, r_k), l \in P_L, r_k \in P_R, k = 1, \ldots, n$. A patch $l \in P_L$ is said to have a match if $S_l \geqq \rho$, where ρ is threshold value (0.75) empirically determined using

a histogram of scores for matching pairs of patches. The patch similarity score between *L* and *R* is defined as:

$$S_{LR}^{p} = \frac{L_m}{|P_L|} \tag{4.11}$$

where L_m is the number of patches in *L* with matching patches in *R*, and $|P_L|$ is the total number of patches in *L*.

Minutiae Patch Relation (φ)

The relationship φ between the pixels around the minutiae in the probe and gallery minutiae patches carry relevant information that assists in validating the minutiae match scores from MDMN. We learn φ between the probe and the gallery minutiae patches from their feature maps obtained from MDMN [12]. Let *p*, *g* represent the probe and gallery minutiae patches, respectively. The relationship between the pixels around the minutiae in *p* and *g* is computed with a neighborhood difference CNN layer following a process similar to that in [12], but with filter size of 3x3 and stride 1. The result is passed through a ReLu layer. From experiments, we found that for matching minutiae pairs, the neighborhood difference *D* is less than 0.25. This threshold was determined from a histogram of *D* shown in Figure 4.4. We define the relation φ between two minutiae patches P_1 and P_2 as :

$$\varphi = \eta D(P_1, P_2) \tag{4.12}$$

where *D* is the neighborhood difference between patches P_1 and P_2, and η = 1 if $D < 0.25$, and 0, otherwise. In the definition of φ, we use η to penalize minutiae scores for patch pairs whose *D* is greater that the threshold value (0.25). We explored another approach (called pixel based approach) for computing φ such as pixel intensity differences between pixels around the minutiae

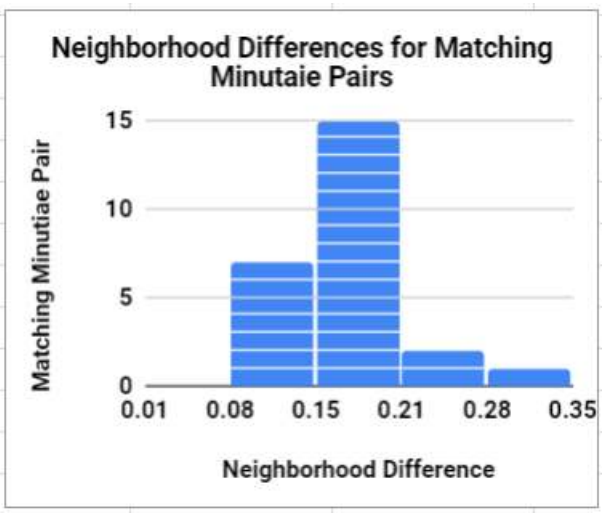


Figure 4.4: Histogram of neighborhood difference (D) of 25 matching minutiae pairs. As can be seen from the histogram, majority of values of neighborhood differences are between 0.08 and 0.21. Setting the threshold to 0.25 for determining matching minutiae pairs worked well in our experiments.

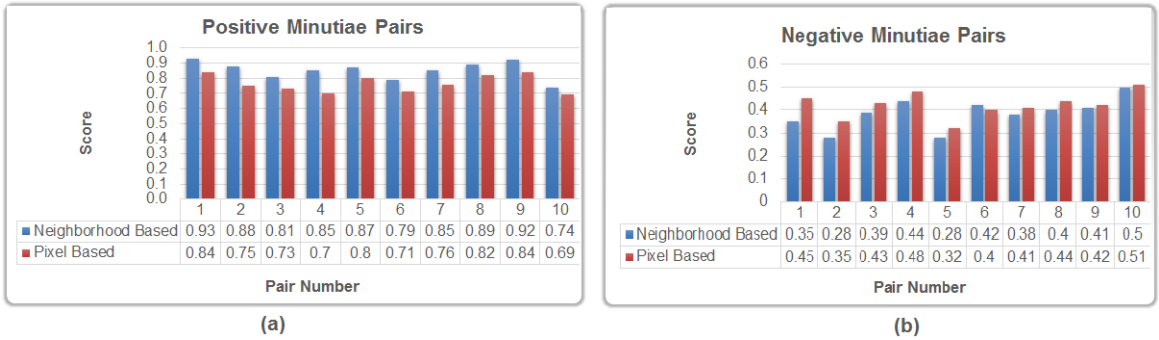


Figure 4.5: Scores for 10 positive and 10 negative pairs of minutiae patches computed using neighborhood and pixel based spatial relationship learning approaches. (a) shows the comparison for positive pairs (higher is better). (b) shows the comparison for negative pairs (lower is better). The neighborhood approach outperforms the pixel approach in both (a) and (b).

and the result we obtained was inferior to that of neighborhood difference approach. We show the comparison of the results obtained using the two approaches in Figure 4.5. It is clear from Figure 4.5 that using the neighborhood difference approach in learning of the spatial relationship between pairs of minutiae patches leads to better results.

Minutiae Detection and Matching

High quality NIST SD4 fingerprint images with NIST Fingerprint Image Quality (NFIQ) values of 1, 2 or 3 were selected for creating the ground-truth patch dataset for training, validation, and testing of the minutiae detection and matching network. The minutiae from the fingerprint images were extracted using open source minutiae extractor MINDTCT from NIST [142]. Patches centered at the minutiae points in each image were extracted from NIST SD4 [8] database and labeled as minutiae patches (mp). Equal number of non-minutiae patches were extracted from the same database and labeled as non-minutiae patches (*nmp*). The *mp* and *nmp* patches were augmented with rotated patches (8 rotations each) and the resulting dataset was normalized to zero mean and unit standard deviation. The dataset was split into three: 80% for training, 10% for validation and 10% for testing the MDMN. There was no overlap between the training, validation and testing datasets. Examples of high quality minutia patches used to train the MDMN are shown in Figure 4.6. Figure 4.7 shows sample fingerprints with minutia quality determined with MINDTCT minutia quality assessment software. Sample images from NIST SD4 database with annotated minutiae are shown in Figure 4.8.

For single-step matching, the minutiae matching score $\rho = s_m$, where s_m is the output of the MDMN, in the range $[0, 1]$. For two-step matching, $\rho = s_m + \varphi$, where φ, is the relationship between the pixels around the minutiae in the probe and gallery minutiae patches. To use the MDMN for minutiae detection, we remove the second branch of the MDMN (by removing all its layers). Figure 4.9(a) shows a branch of the trained MDMN used for minutiae detection, while Figure 4.9(b) are two branches of the MDMN used for minutiae matching. Figure 4.10 is the scatter plot of the feature representations of the embedding of the MDMN showing the separation of the

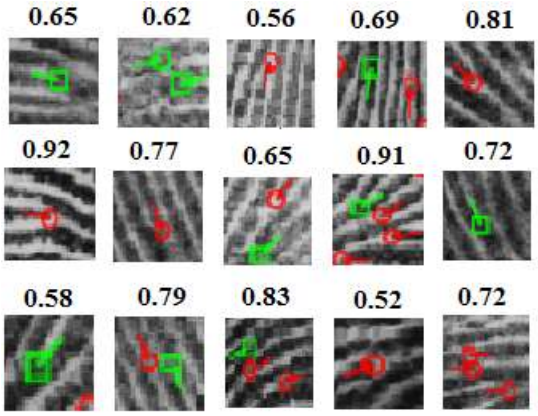


Figure 4.6: Examples of minutia and non-minutiae patches used to train the minutiae detection neural network. Some patches have multiple minutiae. The score for each patch as determined by the MDMN is also shown above the patch. Bifurcations and Ridge endings are annotated in red and green, respectively.

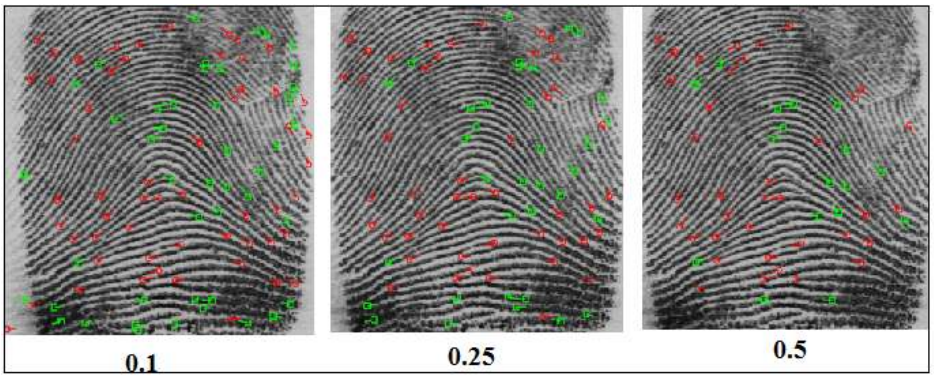


Figure 4.7: Sample fingerprints showing different minutia quality from MINDTCT minutia quality assessment. The quality of each fingerprint is shown below it. Bifurcations and Ridge endings are annotated in red and green, respectively.

embeddings of minutiae and non-minutiae patches.

We define the minutiae match score for each $m \in P_L$ as:

$$S_m = max(\rho_k, \ldots, s(m, r_n)) \tag{4.13}$$

where ρ_k is the minutiae match score of tuple $(m, r_k), m \in P_L, r_k \in P_R, k = 1, \ldots, n$. A patch

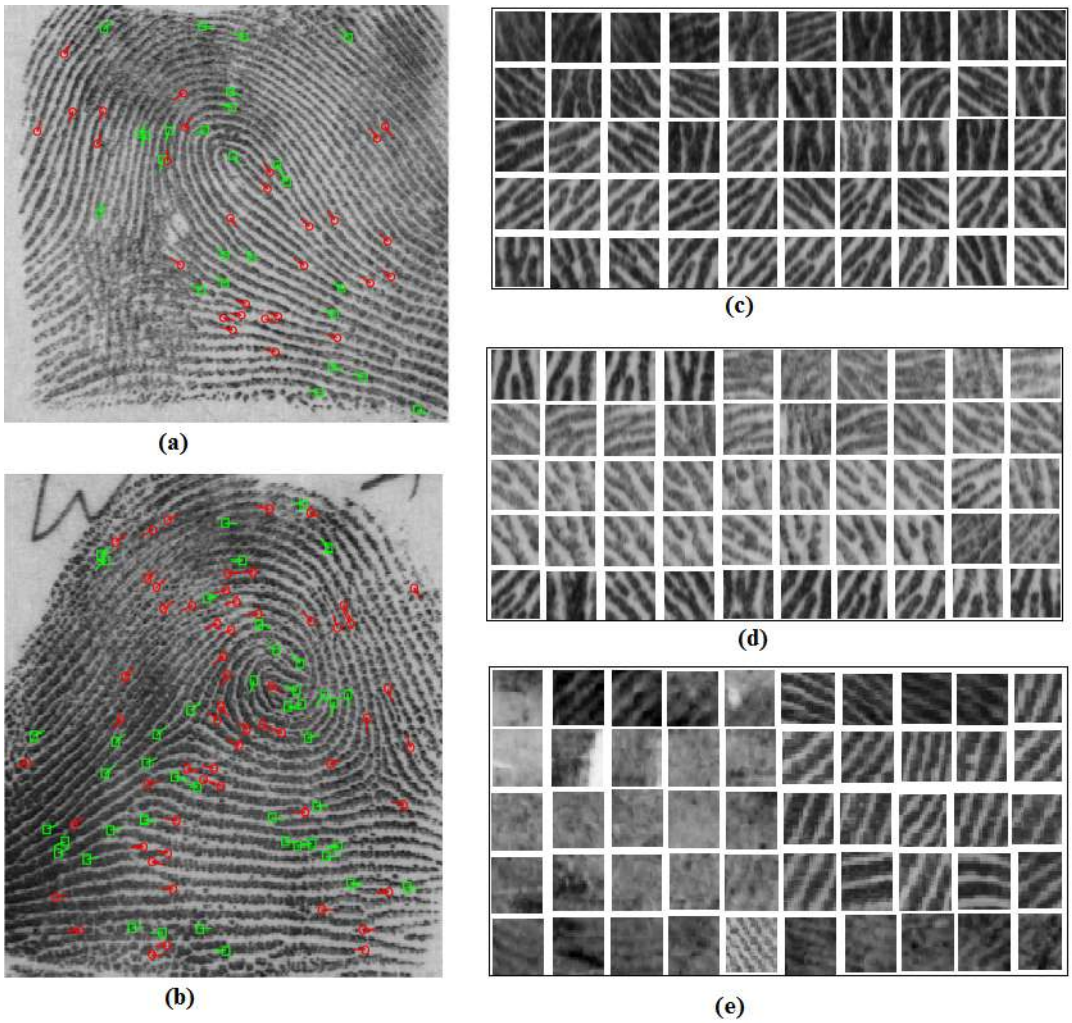


Figure 4.8: (a) and (b) are sample images from NIST SD4 database with annotated minutiae. (c) and (d) are sample 32x32 minutiae patches extracted from NIST SD4 images with centroid at the annotated minutiae. Only minutiae with quality > 0.60 (assessed with MINDTCT) were extracted from the images. (e) shows representative 32x32 non-minutiae patches that were also used in training and validating the MDMN. In (a) and (b), bifurcations are annotated in red and ridge endings are annotated in green.

$m \in P_L$ is said to have a match if $S_m \geqq \psi$, where ψ is an empirically determined threshold value (0.55). The selection of the threshold was guided by the histogram of scores for matching minutiae pairs shown in Figure 4.11.

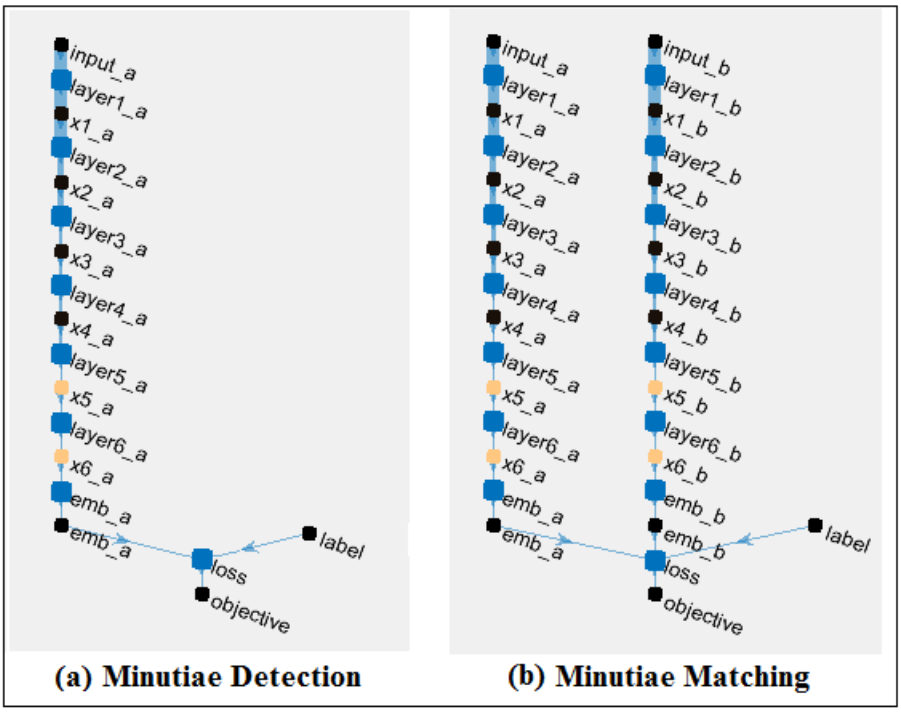


Figure 4.9: Simplified structure of the MDMN showing minutiae detection and matching workflows: (a) is a branch of the MDMN used for minutiae detection, (b) two branches of the MDMN used for minutiae matching.

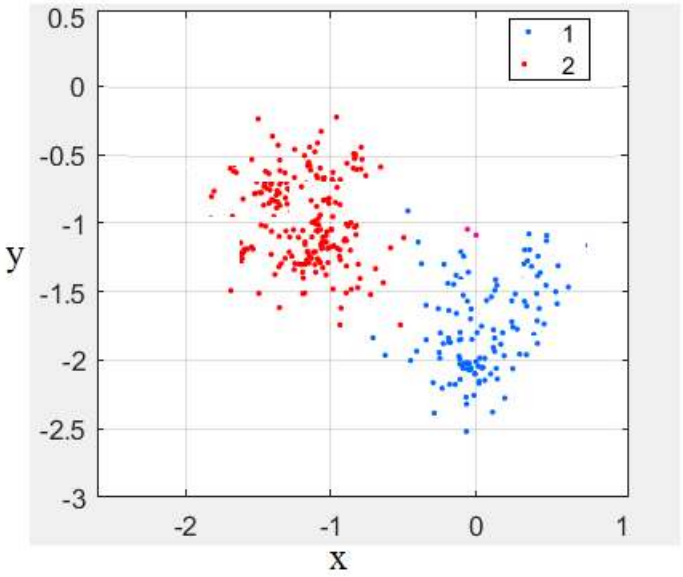


Figure 4.10: Scatter plot for the feature representations of the embedding of the minutiae detection and matching network (MDMN). The representations are for the embedding for a subset of the minutiae detection test data evaluated on the trained embedding part of the MDMN shown in Figure 4.9. Blue (1) is for minutiae patches while red (2), is for non-minutiae patches. The separation of the embeddings is clear.

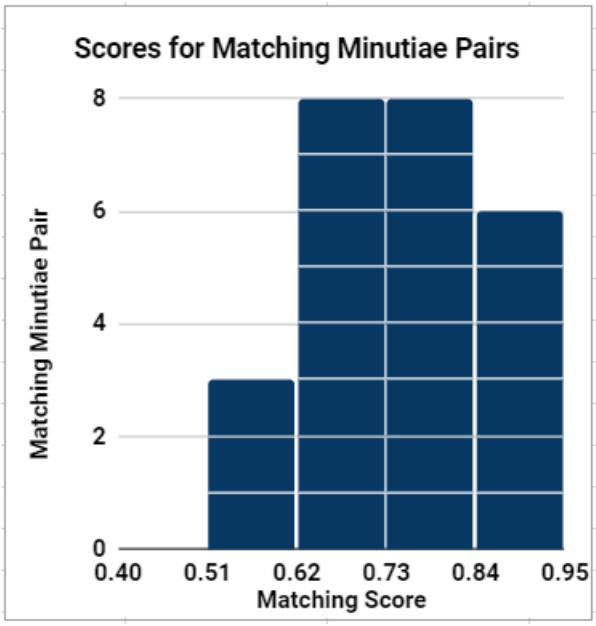


Figure 4.11: Histogram of matching scores of 25 matching minutiae pairs. As can be seen from the histogram, majority of values of matching scores are between 0.51 and 0.95. We set the threshold for determining a match between minutiae pairs to 0.59 and obtained good results.

The minutiae match score between *L* and *R* is defined as:

$$S_M^p = \frac{\varrho}{\gamma} \tag{4.14}$$

where ϱ is the number of minutiae patches in *L* with matching minutiae patches in *R*, and γ is the total number of minutiae patches in *L*.

Fused Matching Score

The fused score is used to make a match/no-match determination between a probe latent fingerprint and a gallery fingerprint. For single-step matching M_p^1 and two-step matching M_p^2, the fused scores are given by:

$$F_{mp}^1 = \frac{1}{2}(S_M^p + S_{LR}^p) \tag{4.15}$$

$$F_{mp}^2 = \frac{1}{3}(S_M^p + S_{LR}^p + \vartheta(F)) \tag{4.16}$$

where S_M^p is the minutiae score, S_{LR}^p is the similarity score, *F* is the difference in spatial frequency between the probe and gallery candidates aggregated over their 32x32 patches, and ϑ is a function that ensures that *F* is in the range $[0, 1]$, defined as:

$$\vartheta = \begin{cases} 1, & \text{if } 0 \leq F < 1 \\ 0, & \text{otherwise} \end{cases}$$

Empirical thresholds of 0.475, and 0.455 determined from the histogram of scores (using Equations 4.15 and 4.16) of 25 matching minutiae pairs and 25 matching non-minutiae pairs, were used for F_{mp}^1 and F_{mp}^2, respectively. Figure 4.12 shows examples of matching patches with minutiae and matching patches without minutiae, while Figure 4.13 shows patch similarity scores computed for 10 segmented latent fingerprints.

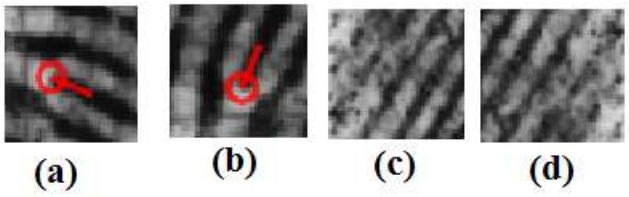


Figure 4.12: Examples of matching query and reference fingerprint patches. (a) and (b) are matching patches with minutiae, while (c) and (d) are matching patches without minutiae. (d) is a 180^o rotation of (c).

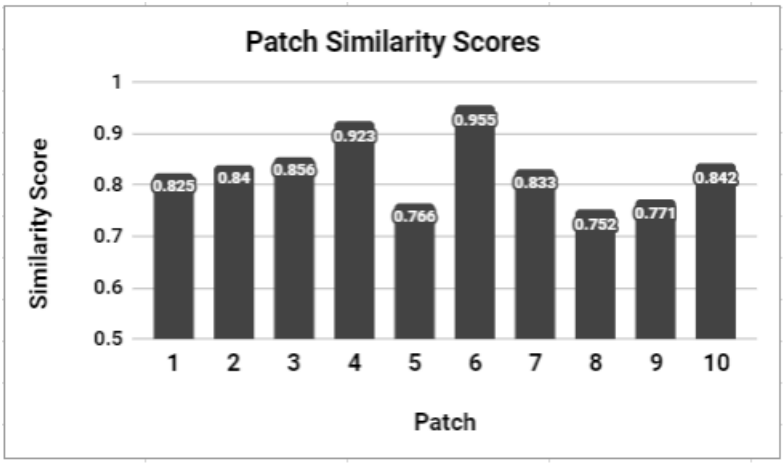


Figure 4.13: Sample patch similarity scores for 10 segmented latent fingerprints from NIST SD27.

4.3 Experiments and Results

We implemented the algorithms in Matlab R2017a running on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. Our implementation relied on Matlab toolbox for neural networks.

4.3.1 Training Validation and Testing

Training Dataset

The training dataset was created from fingerprint images in the NIST SD4 database. The 2,000 images in the database were partitioned into two sets, with 400 in the first set, and 1,600 in

the second set. We split each fingerprint image into 32x32 non-overlapping patches and created 7 similar patches for each patch say p_i by rotating it by $45^0, 90^0, 135^0, 180^0, 225^0, 270^0$, and 315^0. The training dataset was then partitioned into groups of (p_i, p_j, z) where $z = 1$ if $p_i \asymp p_j$ and 0 otherwise. The operator $\asymp$ reflects patch similarity as defined in section 4.2.2. A total of 102,400 patches ($\frac{512\times512\times400}{32\times32}$) from the first set were augmented with the rotated patches to obtain a dataset containing $102,400 \times 8 = 819,200$ patches for training, validating and testing the SLN. Another dataset of 819,200 consisting of 102,400 minutiae and non-minutiae patches (in equal proportions) from the second set augmented with the rotated patches $(102,400 \times 8)$ was used to train, validate and test the MDMN. Each dataset was partitioned as follows: 80% (655,380) patches for training, 10% (81,920) patches for validation, and 10% (81,920) patches for testing.

Evaluation Datasets

We created two evaluation datasets: query dataset and reference dataset. Regions-of-interest were segmented from the 258 latent fingerprints in NIST SD27 database and saved in a query dataset. The segmentation was done using a different deep learning model [55], details of which are omitted for brevity. The reference dataset containing 29,257 rolled prints was created from the 257 rolled images in NIST SD27 database, and 29,000 synthetic fingerprints. Synthetic images were used to boost the size of the background database due to the unavailability of NIST SD14 database [6]. Synthetic fingerprints have been shown to be very useful for training and testing purposes, and have been used for technology evaluations [2]. The synthetic fingerprint images were generated using SFinGe (Synthetic Fingerprint Generator) [10]. We matched each fingerprint in the query dataset against the images in the reference dataset to evaluate the proposed approach.

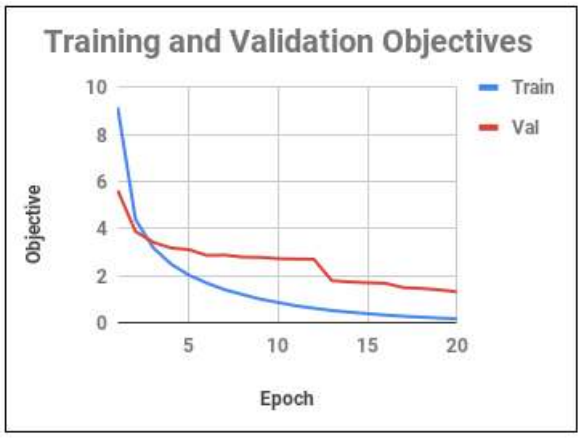


Figure 4.14: Plots of training objective function during training and validation for 20 epochs. The objective is on the y-axis while the training epoch is on x-axis.

MDMN: Training, Validating and Testing

To train the MDMN network, one set of fingerprint patches at a time from the training dataset was fed to the network after being transformed to the frequency domain and preprocessed to have 0 mean and unit standard deviation. The validation data was fed through the forward pass of the network and was used to check the response of the model being trained to data that it was not currently being trained on. Training the MDMN involved minimizing the objective defined by Equation 4.2. Figure 4.14 shows plot of minimization of the training objective during training and validation of the MDMN.

SLN: Training, Validating and Testing

Training of the SLN involved the minimization the cross-entropy error defined as:

$$C = -\frac{1}{m}\sum_{k=1}^{m} y_k \log \hat{y}_k + (1 - y_k)\log(1 - \hat{y}_k) \tag{4.17}$$

over a training set of *m* patch pairs using stochastic gradient descent with a batch size of 64. In the above equation, y_k, is the similar (1) or dissimilar (0) label for input pair x_k, while $\hat{y}_k$ and $1 - \hat{y}_k$

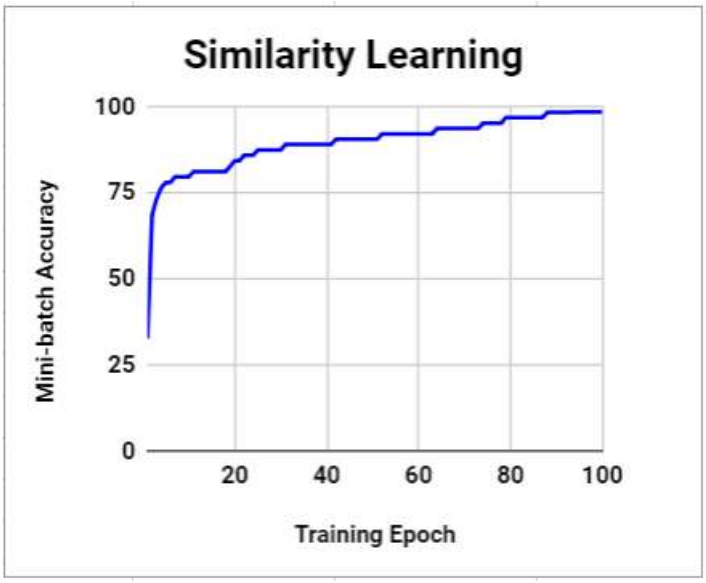


Figure 4.15: Plot showing mini-batch accuracy during the training of the similarity learning network (SLN). The training was done for 100 epochs.

are the Softmax activations computed on the values at the two output nodes.

For each network, we sampled equal number of positive and negative pairs for training to prevent over-fitting. A positive entry in each training dataset is of the form $(p_i, p_k, 1)$, with $p_i = p_k$, while a negative sample is of the form $(p_i, p_j, 0)$, with $p_i \neq p_j$. We used equal number of positive pairs and negative pairs to train, validate and test each network. There was no overlap between the training, validation and test subsets. For SLN, we obtained 97.21%, 95.73%, and 94.15% training, validation and testing accuracy, respectively. For MDMN, the results were 97.25%, 94.67%, and 92.38% for training, validation and testing, respectively.

4.3.2 Frequency Domain vs. Spatial Domain

To evaluate the benefit of using frequency domain data representation for training both the MDMN and SLN, we also trained the networks with spatial domain representation of the same

training dataset. In the first experiment, we used the spatial domain representation of the image patches to train and validate both networks. In the second experiment, both networks were trained and validated using the frequency domain representation of the patches obtained by taking magnitude of the DFT coefficients of the patches in the dataset. Figure 4.17 shows the training and validation results. Clearly, the models trained with frequency domain representations of the training dataset achieved higher training and validation accuracy as well as lower training and validation loss compared when they were trained with spatial domain representation of the training datasets. This indicates that features learned from the frequency domain transformations of the patches are more discriminative than those learned from raw patches.

4.3.3 Performance Evaluation Metrics

We used the following metrics to evaluate the performance of our network.

Receiver Operating Characteristic (ROC) Curve

ROC curve provides a means of comparing the performance of a set of classification models or mutations of a classification model. It shows false positive rate (1-specificity) on the X-axis, against true positive rate (sensitivity) on Y-axis. The false positive rate is the probability of target being true when its true value is false. True positive rate is the probability of target being true when its true value is true. For a good performing model, the curve climbs quickly towards the top-left of the chart. This indicates that predictions from the mode are correct in most cases.

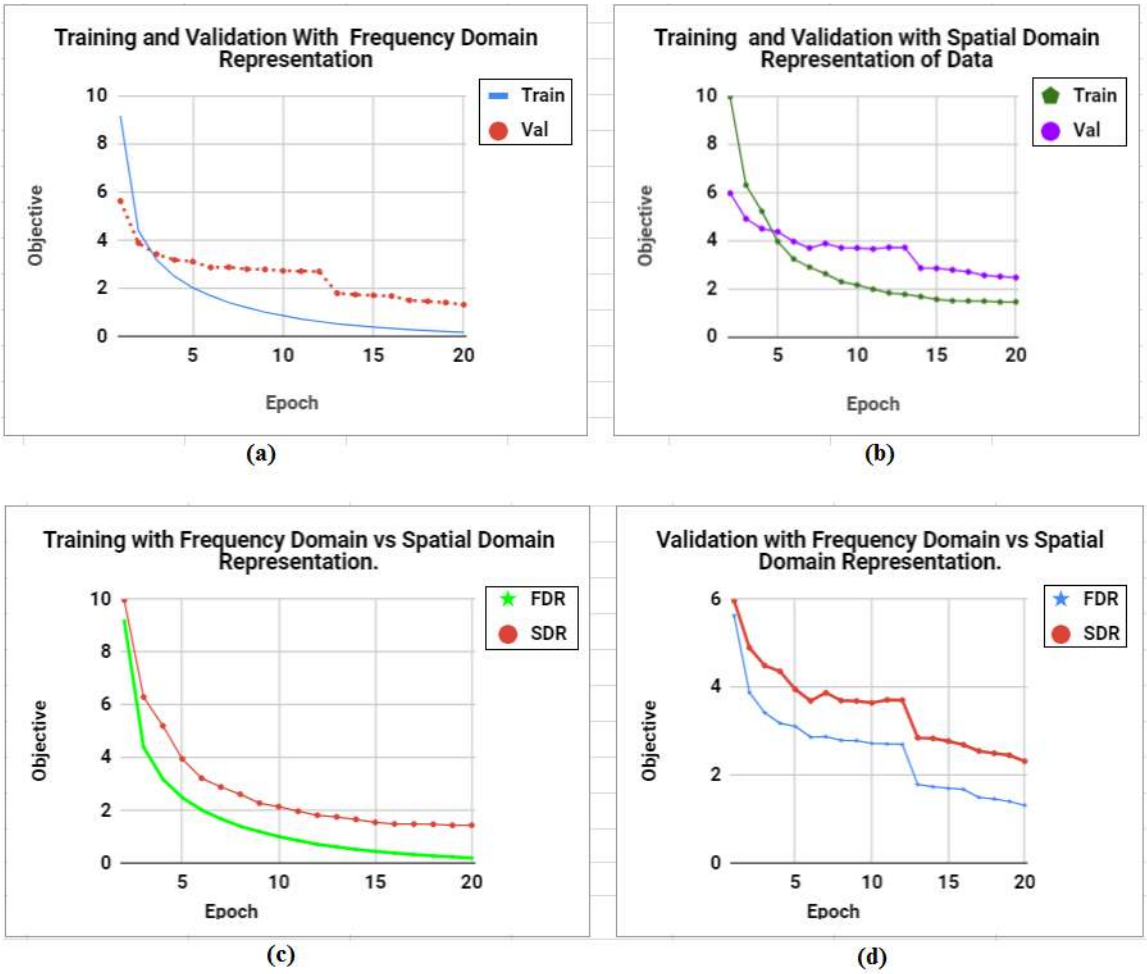


Figure 4.16: MDMN training and validation performance with frequency domain representation (FDR) vs. spatial domain representation (SDR) of the training dataset. (a) shows the training and validation loss when the model was trained and validated with FDR of the training and validation datasets, while (b) shows the training and validation loss when it was trained and validated with SDR training and validation datasets. From the plots, it is clear that better performance is achieved when the model is trained with frequency domain representation of the training and validation datasets.

Area Under ROC Curve (AUC)

AUC is used as a measure of quality of the classification models. An AUC of 0.5 indicated a random classifier, while AUC of 1 indicates a perfect classifier. Most practical classification models have AUC between 0.5 and 1.

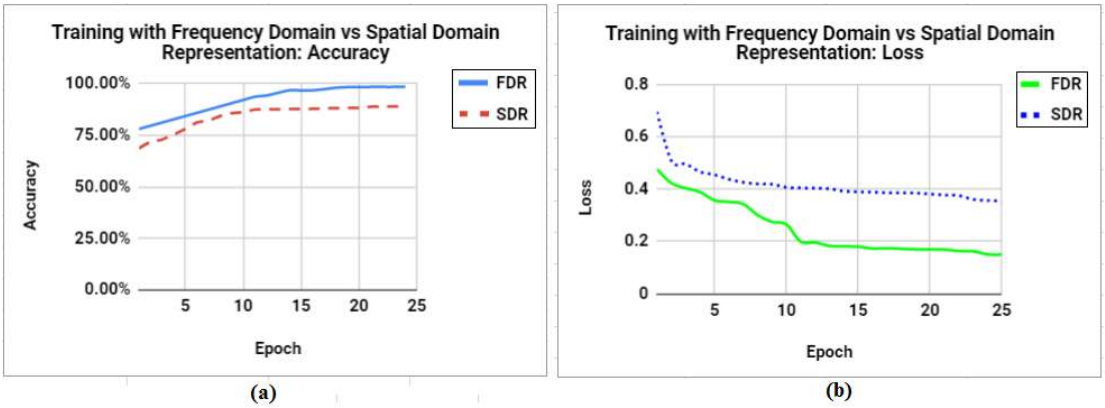


Figure 4.17: SLN training and validation performance with frequency domain representation (FDR) vs. spatial domain representation (SDR) of the training dataset. (a) shows the training and validation loss when the model was trained and validated with FDR of the training and validation datasets, while (b) shows the training and validation loss when it was trained and validated with SDR training and validation datasets. From the plots, it is clear that better performance is achieved when the model is trained with frequency domain representation of the training and validation datasets.

Cumulative Match Characteristics (CMC)

CMC is a method of summarizing the measured performance of a biometric system in a closed-set identification setting. Biometric probe and gallery candidates are compared and ranked based on their similarity/matching scores. CMC shows how often the biometric probe candidate appears in the ranks (1, 5, 10, 20, etc.), based on the match rate and provides a way for comparing the rank against the identification rate.

4.3.4 Matching Results and Comparison

Rank identification rate provides an estimate of the probability that a matching rolled fingerprint is identified correctly at least at rank-*k* during a search with a latent candidate. Figure 4.18(a) shows the cumulative match characteristics (CMC) curve of the proposed approach in matching 258 latent fingerprints in NIST SD27 database against database of 29,257 fingerprints consisting of the 257 rolled images in NIST SD27, and 29,000 synthetic fingerprints generated with

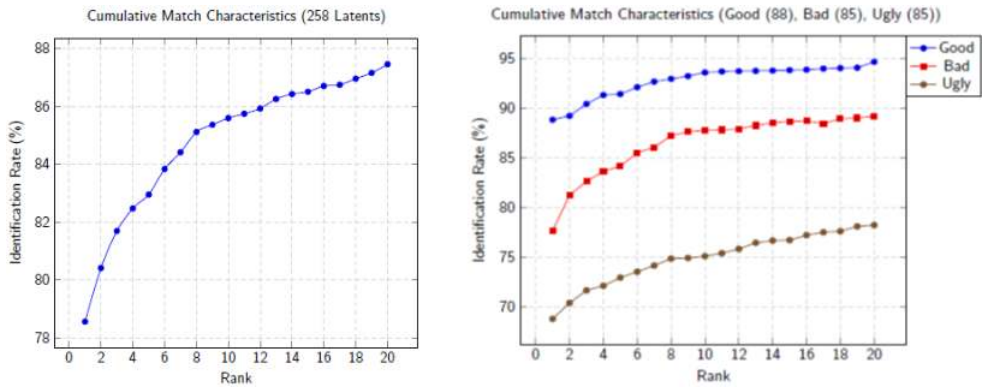


Figure 4.18: (a) CMC curve of the proposed approach in matching 258 latent fingerprints in NIST SD27 database against a test database of 29,257 rolled fingerprints. (b) CMC curves for the 258 latent fingerprints in the NIST SD27 database that were grouped by subjective quality into Good (88), Bad (85), and Ugly (85). These results were obtained by running the matching tests 10 times and averaging the results.

SFinGe [10]. Figure 4.18(b) shows CMCs of matching the three categories of latent fingerprints in the NIST SD27 database (88 Good, 85 Bad, and 85 Ugly) [74] against a database of 29,257 fingerprints. The plot shows the rank-k identification rate against k, $k = 1, \ldots, 20$. We obtained a rank-1 identification rate of 78.56% and a rank-20 identification rate of 87.45% on matching the 258 latent fingerprints. These results look promising when compared to a state-of-the-art rank-1 and rank-20 identification rates of 74.0% and 82.9%, respectively, reported in [75], which to the best of our knowledge, is the state-of-the-art result. Table 4.1 shows the rank-1 and rank-20 identification rate comparison with recent latent fingerprint matching algorithms. There is no guarantee that the results we obtained using a reference database augmented with synthetic fingerprints will match the results we would obtain using a reference database of the same size with real fingerprints, but the results indicate the likely performance of our model in a realistic setting. It should also be noted that synthetic fingerprints generated using SFinGe have been used in many fingerprint verification competitions with results obtained comparable to results obtained with real fingerprint databases [33].

Author	**Good (%)**	**Bad** (%)	**Ugly** (%)
Verifinger	75.0	47.0	30.6
Feng et al. [60]	78.4	55.3	52.9
This work	**88.84**	**77.67**	**68.82**

Table 4.4: Rank-1 accuracies of the proposed matcher, verifinger and Feng et al. [60] on Good, Bad and Ugly categories in NIST SD27.

Table 4.4 show the comparison of the accuracy of the proposed matcher to the published matching results on the Good, Bad and Ugly categories of NIST SD27. The proposed matcher achieves better results in the three categories.

Image Quality versus Single-step and Two-step matching

Equations (4.8) and (4.9) define Single-step and Two-step matching, respectively. We investigate the best combination of features (Correlation, Minutia, Relation and Spatial frequency) defined in Section 4.2.2, that yields the best DeepLatent matching performance for a given image quality. Experiments were performed using the following combination of features: (a) Correlation only, (b) Minutiae only , (c) Correlation, and Minutiae, (d) Correlation, Minutia, and Relation, and (e) Correlation, Minutia, Relation and Spatial frequency. We ran five experiments using five different datasets. Each dataset contained patches with the same NFIQ number $i, i = 1, \ldots 5$. Each experiment covered (a) to (e) above. For each experiment, we obtain Receiver Operating Characteristic (ROC) curve with probability of true match P_t vs. probability of false match P_f, defined as:

$P_t = P\{\text{true match} \mid \text{potential match is } \textit{a true} \text{ match}\}$,

Experiment	Quality	NFIQ	M only	C only	C and M	C, M and R	C, M, R and F
1	Excellent	1	0.5243	0.7783	0.9596	0.9745	0.9875
2	Very Good	2	0.5353	0.7550	0.9648	0.9697	0.9826
3	Good	3	0.5405	0.7529	0.9340	0.9559	0.9605
4	Fair	4	0.4239	0.6067	0.8120	0.8302	0.8423
5	Poor	5	0.3765	0.5257	0.7410	0.7752	0.7888

Table 4.5: Area under ROC curves (AUCs) for Figure 4.19 showing the justification for using single-step or two-step matching based on image quality. Each row shows the experiment number, the NFIQ number for the patches in the dataset used for the experiment, and the AUCs for the various feature combinations. When the image quality is excellent or very good, using a combination of minutiae (M), correlation (C), relation (R) and frequency (F) features gives slightly better AUC (0.9875), than using minutiae and correlation (0.9596). However, the computational burden due to the computations required for R and F (Table 4.6, column 3) may outweigh the performance gain. For excellent and very good quality images (NFIQ numbers 1, 2), single-step matching that uses minutiae and correlation features should be used. The AUCs for images with NFIQ numbers 3, 4, 5 (Good, Fair, Poor) show that two-step matching that uses all the four features gives better matching results than single-step matching when the NFIQ number of the latent fingerprint image is greater than 2.

$P_f = P\{\text{true match} \mid \text{potential match is } \textit{not a true} \text{ match}\}$

Figure 4.19 shows the matching performance of DeepLatent in various scenarios. The Area Under Curve (AUC) for the experiments are summarized in Table 4.5 The results confirm that DeepLatent performs better with better quality image patches. It also shows that using minutiae or correlation alone does not give good matching results. It should be noted that although using a combination of minutiae, correlation, relation and spatial frequency features with good quality image gives slightly better matching results, the additional computational burden shown in Table 4.6 may outweigh the performance gain. For good quality images, single-step matching that uses minutiae and correlation features should be used. For moderate to low quality images, two-step matching that uses all the four features should be used.

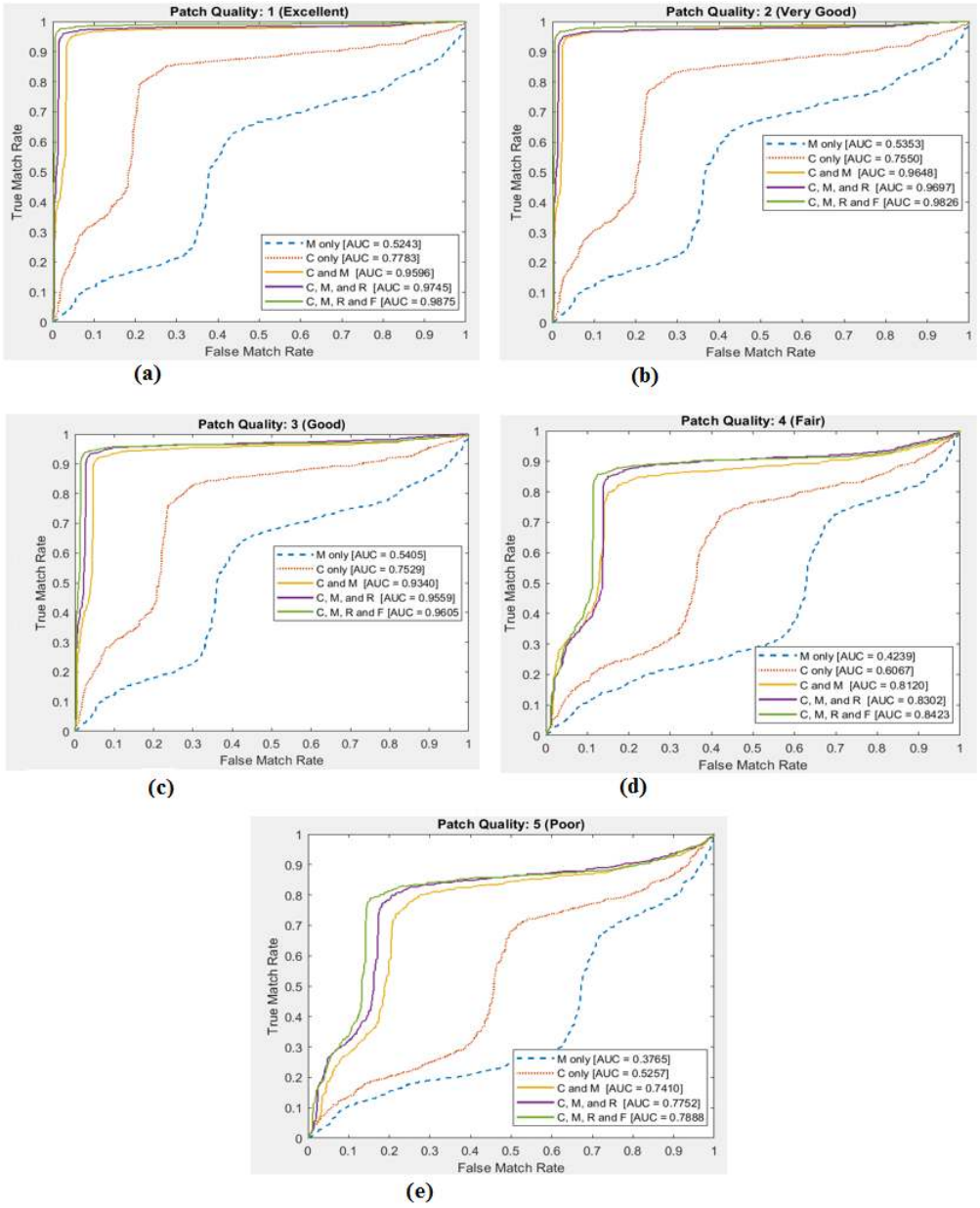


Figure 4.19: ROC curves for image quality vs single-step/two-step matching experiments. Each ROC is for one NFIQ number (1, 2, 3, 4, 5). The AUCs for the various feature combinations are shown. The results show that DeepLatent performs better with better quality image patches. Also using a combination of minutiae, correlation, relation and frequency yields better performance than using just minutiae and correlation. Table 4.5 provides more information on the AUCs.

FDR and SLN Performance

We investigated the accuracy of matching patches based on the similarity of spatial domain representations (SDR) of the patches versus similarity of their frequency domain represen-

Experiment	Single-Step	Two-step	R and F Computation
1	1,876.45	2,405.18	528.73
2	1,791.28	2,394.26	602.98
3	1,885.19	2,471.88	586.69
4	1,902.65	2,428.27	525.62
5	1,795.73	2,426.13	630.40

Table 4.6: Computational cost (in seconds) for Figure 4.19 showing the justification for using Single-step matching when the image quality is excellent or very good (NFIQ number 1 or 2).

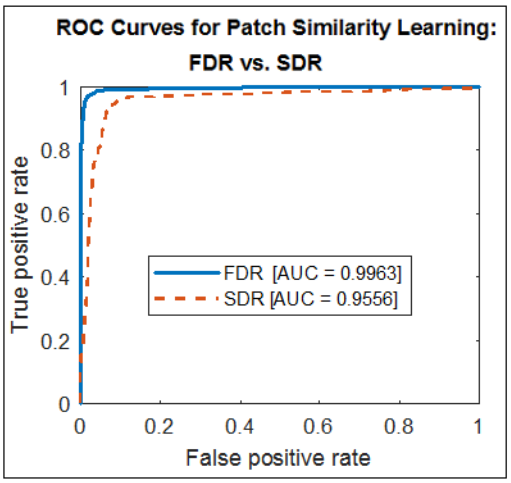


Figure 4.20: ROC curves for similarity measures with raw image patches and frequency domain representations of image patches. Dotted line is for raw image patches and solid line is for frequency domain representations.

tations (FDR). We use ROC curves to compare the performance of SLN (Figure 4.20) using raw image patches and frequency domain representation for similarity learning. In this experiment, we are interested in the accuracy of matching similar patches. A comparison of the two curves (dotted for raw patches and solid for frequency domain representations) shows that for similarity learning, using frequency domain representation is more effective than using spatial domain representation.

Dimension	Computation (sec.)	Mean Accuracy	Mean Loss
512	57.66	0.7146	0.6433
1024	101.25	0.7918	0.6213
2048	250.19	0.8156	0.5861
4096	850.74	0.8921	0.5132
8192	1,230.44	0.9145	0.4321
16384	1,962.60	0.9547	0.214
23520	**2,786.22**	**0.9762**	**0.136**
32760	3,524.54	0.9771	0.131
65520	8,145.42	0.9795	0.128

Table 4.7: Tradeoff: Accuracy vs. feature dimension. Training was done for 100 epochs. The model with 32,732 fully connected layer size performed slightly better than the one with 23,520 nodes. However, the 0.0009 performance gain is not worth the additional computational cost (738.32 seconds). We set the feature dimension of the SLN to 23,520 because it gave the best combination of computation cost, accuracy and loss.

Feature Dimension: Accuracy vs. Computation Time Tradeoff

We evaluated five variations of the SLN architecture to study the impact of feature dimensionality on the accuracy and computational burden of the similarity learning network, by adjusting the sizes of the fully connected layers in the two networks. Table 4.7 shows the size of the fully connected layer (feature dimension), with the associated computational cost, accuracy and loss. When the size of the fully connected layer was increased to 32,732 nodes, the model achieved 0.9771 accuracy but took 3,524.54 second to converge. We found an optimal fully connected size of 23,520 with model accuracy of 0.9762 and 2,786.22s to achieve convergence. The 738.32 seconds difference in the computation time (3,524.5 - 2,786.22 = 738.32) is not commensurate with the 0.0009 (0.9771 - 0.9762) performance gain in accuracy.

DeepLatent Computational Cost

The MDMN network took around 8-10 hours to converge. The SLN network took 4.5 - 5.5 hours to converge. Both networks were trained on a system with Intel Core i7 CPU and 8GB RAM. The average processing time for a single ROI input image, including minutiae detection, patch extraction, and computation of the matching score using the trained model is 850ms. On the average, matching a NIST SD27 latent ROI against top 200 gallery candidates takes about 15.2 seconds. Given that matching accuracy is of utmost importance in latent fingerprint matching, the accuracy of our matching algorithm is worth the computational cost. The overall matching computational cost can be improved by running multiple matching pipelines and using distributed processing for large scale deployment.

Robustness to Rotation

Robustness to rotation is built into DeepLatent by training and validating the model with rotated and non-rotated patches. To verify this assertion, we tested the trained DeepLatent with 6,128 32x32 patches consisting 1,328 probe patches from segmented [55] ROI latent fingerprint and 4,800 gallery patches from a matching ten-print (gallery). The probe and gallery images are from NIST SD27. Table 4.8 shows the details of the probe and gallery images used for this robustness to rotation experiments. Four tests were done with the four datasets shown in Table 4.8. In the test with dataset 1, the probe and gallery patches were not rotated. In test 2 the gallery patches were rotated, and in test 3, the probe patches were rotated. In the last test, we rotated both gallery and probe patches. Five experiments each involving four tests with the four datasets were performed using the segmented ROI from the following latent fingerprints (Good, Bad, Ugly) from

	Segmented Probe Image	**Gallery Image**
Size	380x448	800x768
32x32 patches	$\frac{380*448}{1,024} = 166$	$\frac{800*768}{1,024} = 600$
Rotations	8	8
Total 32x32 patches (with rotations)	166 *8 = 1,328	600 * 8 = 4,800
Dataset 1	166	600
Dataset 2	166	4,800
Dataset 3	1,328	600
Dataset 4	1,328	4,800

Table 4.8: Robustness to rotation. Patches were rotated $\{0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o\}$.

NIST SD27 and their rolled ten-print mates: (G068L6, G068T6), (G065L6, G065T6), (G055L3, G055T3), (B129L7, B129T7), (U300L2,U300T2). We performed experiment 1 with probe ROI and gallery pair (G068L6,G068T6), experiment 2 with (G065L6, G065T6), experiment 3 with (U300L2, U300T2), experiment 4 with (B129L7, B129T7), and experiment 5 with (G055L3, G055T3). The matching scores shown in Figure 4.21 clearly support our assertion that DeepLatent matching performance is robust to rotation. It is also worthy to note that the better the quality of the probe image, the better the matching score from DeepLatent.

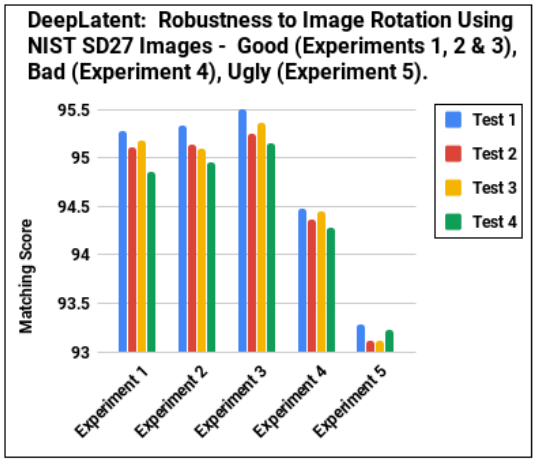


Figure 4.21: Matching performance of DeepLatent using the datasets shown in Table 4.8. The accuracy of DeepLatent in the four tests and five experiments shows the model's robustness to image rotation. The results from these experiments also indicate that the better the quality of the latent fingerprint ROI image, the better the matching score.

4.4 Summary

This chapter proposed a unified frequency domain based framework for latent fingerprint matching using image patches. The matching is based on the similarity of the frequency domain representations of patches encoded in a deep neural network, and the minutiae on the correlated patches. For minutiae detection and matching, we presented a Siamese hybrid network with a neighborhood relationship computation and summarization layer for determining how similar two minutiae patches are based on the relationship between the pixels in the neighborhood of minutiae. The chapter also presented empirical discussions on the performance of the proposed model when trained with frequency domain representation versus spatial domain representation of fingerprint image patches. The proposed system was tested by matching segmented fingerprints from 258 latent fingerprints in NIST SD27 against a database consisting of 29,257 fingerprints fingerprints and achieved a rank-1 identification rate of 78.56%. This is a significant improvement on the state-of-the-art rank-1 identification rate, which to the best of our knowledge is 74%.

Chapter 5

LAFISR: Latent Fingerprint Image Super-Resolution using Deep Convolutional Neural Networks

The quality of a latent fingerprint image provides an indication as to whether the latent fingerprint is a good candidate for further forensic analysis and feature annotations. In both legal system and forensic science literature, the accuracy of latent fingerprint identification by latent fingerprint forensic examiners has been the subject of increased study, scrutiny, and commentary. When there are errors in latent fingerprint matching, the impact can be devastating, resulting in missed opportunities to apprehend criminals or wrongful convictions of innocent people. As shown in Figure 5.1, Latent fingerprint images contain background structured noise such as stains, lines, arcs, and sometimes text, making it hard to process and extract enough relevant features for matching. Enhancing the quality of latent fingerprint images is essential for effective and reliable match-

ing. This chapter addresses the problem of latent fingerprint image enhancement by generating high resolution latent fingerprint from a low resolution latent fingerprint through a technique referred to as Single Image super-resolution (SISR). SISR is a method for generating a high-resolution image from a low-resolution image by reconstructing the high-frequency components containing details missing from the low-resolution image. It is considered an extension of image restoration [105]. For a given latent fingerprint image, we aim at producing an image of the same size but with significantly higher image quality. We learn a set of filters which when applied to a given latent fingerprint image produces a higher resolution version of it. We embed an image enhancement algorithm in the proposed super-resolution algorithm to generate enhanced SR output image. Using a graph-total variation energy of latent fingerprints as a non-local regularizer for a convolutional neural network, we learn optimal weights for high quality image reconstruction. The latent fingerprint super-resolution problem can be formulated as:

$$l = \beta z + e \tag{5.1}$$

where l is the low-resolution latent, z is the unknown high resolution latent, β is a linear operator that blurs and decimates z, and e represents noise. The goal of the proposed LAFISR algorithm is to find z given l. We write the regularized solution to Equation 5.1 in variational form as:

$$\hat{z} = \arg\min_{z \in \mathbb{R}^n} \frac{1}{2}\|l - \beta(z)\|^2 + \varphi \mathcal{J}(z) \tag{5.2}$$

where $\hat{z}$ is the regularized solution to the SR problem, $\mathcal{J}$ is the energy function that tends to zero when z is close to the smoothness model [107], l is the low resolution image, φ is the weight that needs to be adapted to l, and β is a linear operator that blurs and decimates z. Our goal is to learn a model (LAFISR) that can reconstruct a $\hat{z}$ that is close to z. LAFISR is a deep convolutional neural

network with multiple cascaded layers of convolutional filters. The LAFISR framework aims at recovering $\hat{z}$ from a set of noisy measurements $l = \beta z + e$ using $\mathcal{J}(z)$. To accomplish that goal, we minimize the objective function defined in Equation 5.2 during training.

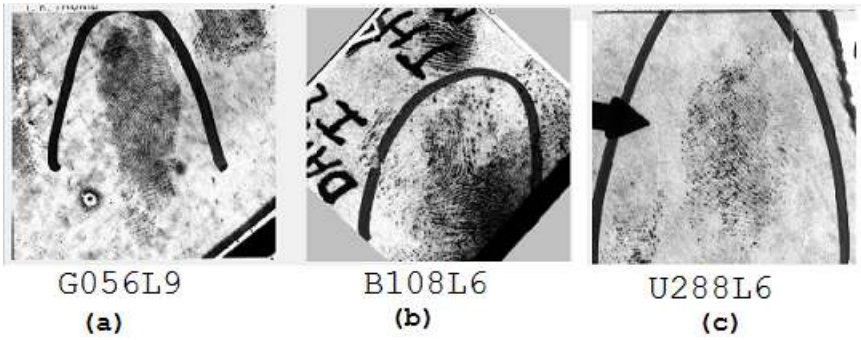


Figure 5.1: Sample latent fingerprints from NIST SD27 showing three different quality levels (a) good, (b) bad, and (c) ugly.

We demonstrate empirically that matching performance can be improved by preprocessing latent fingerprints using our proposed super-resolution model. A comparison with a number of recently published methods for fingerprint enhancement using qualitative and quantitative evaluation metrics show that our model outperforms existing approaches.

The rest of the chapter is organized as follows: In Section 5.1, we present related work and contributions. Technical approach and the proposed network architecture are highlighted in section 5.2. In Section 5.2.1, we present the strategy used in selecting the depth of the network. Patch based regularization is presented in section 5.2.2. The training and evaluation datasets, performance evaluation metrics, quantitative and qualitative evaluation of the proposed model, comparison with other methods, as well empirical analysis of the impact of patch based regularization on the performance of the proposed model are presented in section 5.3. Section 5.4 contains concluding remarks and future research direction.

5.1 Related Work and Contributions

Single image super-resolution (SISR) methods have been developed for image processing and analysis tasks such as face recognition [15], medical imaging [141], and video surveillance [150]. Early SISR methods included bicubic interpolation [61], edge guided interpolation [154], and Lanczos resampling [48]. These linear methods have not fared well in reconstructing complex image structures resulting in aliasing artifacts and over-smoothed regions in the super-resolved image [109]. Recent SISR techniques have learnt mappings from low resolution to high resolution images. Among these are those based on neighbor embedding [37, 36], sparse coding [145, 135, 113], random forest [121, 94], and convolutional neural networks [109, 81, 47, 114, 134].

Sparse coding based methods learn compact dictionaries from sparse signal representations and produce compact representations of pairs of low resolution and high resolution image patches over the learnt dictionaries. Example based methods learn a mapping from low resolution (LR) patches to their high resolution (HR) mates using external database of images. As demonstrated in [81], CNN can be used to learn a mapping from LR to HR in an end-to-end manner, without the requirement of hand-crafted features that are typically necessary in other methods.

Super-resolution methods have also been developed for fingerprints image enhancement [151, 126, 27]. Yuan et al. [151] proposed a super resolution based fingerprint image enhancement using early stopping machine learning as a regularizer, with boundary constraint added to ensure regularity of reconstruction. Bian et al. [27] reconstructed the SR image by using sparse representation with ridge pattern prior based on classification with coupled dictionaries. Singh et al. [126] used ridge orientation-based clustered coupled sparse dictionaries to reconstruct the SR image. They used the minimum residue error criterion for choosing a sub-dictionary for a given

patch, while applying back projection to eliminate the discrepancy in the estimate due to noise or inaccuracy in the sparse representation.

5.1.1 Contributions

In light of the above related work, the contributions of this chapter are:

- A novel latent fingerprint super-resolution method that uses graph-total variation energy (see equation (5.3)) of latent fingerprints as a regularizer and prior for optimal weights of the network . The proposed convolutional neural network directly learns an end-to-end mapping between low-resolution and high-resolution fingerprint images.

- Enhancement of minutiae and ridge structures are performed using the learned filters. Low resolution patches are mapped to their enhanced high resolution versions, leading to increased image resolution and contrast enhancement at the same time.

- Detailed experiments show that super-resolution processing of latent fingerprints achieves good matching performance even with low quality latent fingerprint images.

Unlike [151, 126] and [27], the proposed method is based on deep CNN and uses graph-total variation energy of latent fingerprints as a non-local regularizer for learning optimal weights for high quality image reconstruction. In addition, our model targets latent fingerprints that are more difficult to process than rolled fingerprints considered in [151, 126] and [27]. Using graph-total variation energy of latent fingerprints as a regularizer and prior for optimal weights of the deep CNN distinguishes our model from other CNN based single image super-resolution algorithms presented in [109, 81, 47, 114] and [134].

5.2 Technical Approach

The proposed latent fingerprint super-resolution convolutional network consists of 15 weight layers, 15 ReLu layers, and 1 Regression layer. We use 64 filters each of which is of size 3 x 3 x 64. Each filter operates on 3 x 3 spatial region across 64 channels (feature maps). The configuration of the network is depicted in Figure 5.2. The first layer operates on the input image. The last layer is used for image reconstruction and consists of a single filter of size 3 x 3 x 64. Staring with a bicubic interpolated low-resolution image as input, the network models the details of the input image and predicts a residual image. The super-resolved image is obtained by adding the residual image to the input image. To keep the sizes of all feature maps from shrinking as we go through the layers of the network, zero padding is done before convolutions. Kim et al. [81] exploited contextual information over large image regions by cascading small filters in a deep convolutional neural network. They used effective training procedure that learnt the difference between HR and LR images (residuals) at multiple image scales. As reported in [81], using this strategy enables correct prediction of pixels near image boundaries. The proposed network is inspired by [81] but uses a patch based prior as a regularizer and has 15 weight layers instead of 20 used in [81]. We performed a multi-scale (2, 3, and 4) training of our model, producing a network with super-resolution machines of multiple scales capable of performing super-resolution at multiple scale factors without appreciable performance degradation.

5.2.1 Network Depth

The choice of the network depth of LAFISR was empirically determined by training and testing networks of depth ranging from 5 to 21 weight layers (counting convolutional) or 10 to

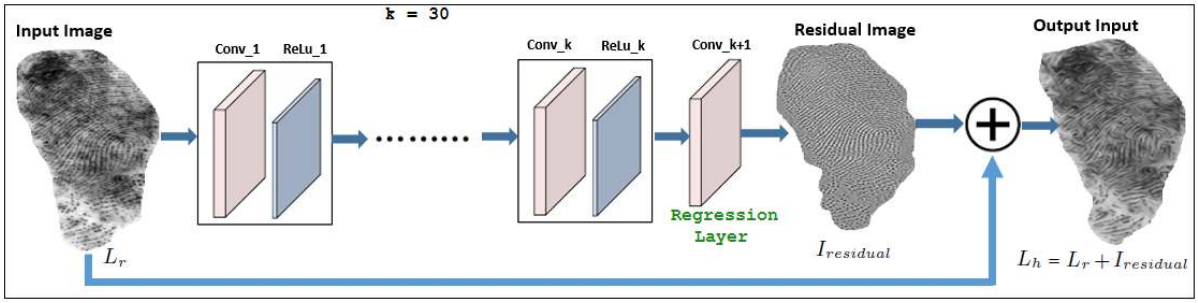


Figure 5.2: Proposed Network consisting of convolutional and ReLU layers cascaded to a depth of 31. A low resolution latent fingerprint L_r is fed to the network. It is transformed into a high-resolution L_h image after passing through the layers of the network. The network predicts a residual image $I_{residual}$ which is added to L_r. The resulting image ($L_r + I_{residual}$) is enhanced by amplifying the structures/details in the latent fingerprint image for reliable feature extraction.

42 (counting both convolutional and nonlinearity layers), using 20 fingerprint images from NIST SD4 database. After each experiment, we increase the depth by 1 for the next experiment. Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Naturalness Image Quality Evaluator (NIQE) defined in section 5.3.1 were used to compare the performance of the various architectures. The plots of PSNR, SSIM and NIQE results for three scale factors (2, 3, 4) are shown in Figure 5.3. For each scale factor, the performance increases rapidly as depth increases, up to a certain depth (15 in our experiment), before flattening out. We also show the PSNR, SSIM, NIQE and computational cost for scale factor 2 for depths 5 to 21 in Table 5.1. Since the minimal gain in performance after depth 15 is not commensurate with the computational time needed for convergence, we chose the network with 15 weight layers.

5.2.2 Patch Based Regularization

Non-linear filtering that is adaptive to image content enables non-local averaging of image features [107]. Such filters are useful in averaging pixels in an image by measuring the distance

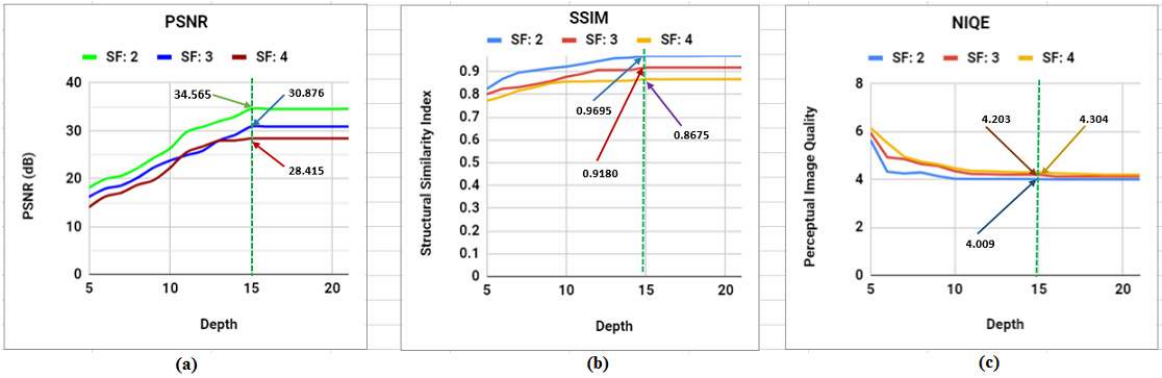


Figure 5.3: LAFISR network depth selection: SF stands for scale factor. The performance in terms of PSNR and SSIM increases rapidly as depth increases, up to depth 15 weight layers (counting only convolutional layers) and depth 30 (counting both convolutional layers and nonlinearity (ReLU) layers), and flattens. The NIQE decreases (less is better) as depth increases and also flattens after a depth of 15 weight layers. For depths 16 through 21 weight layers, the performance gain was minimal and the model took longer to converge. A network with 15 weight layers was chosen because it gave the best performance compared to the other explored networks with lower of higher weight layers.

between image patches. Given that image super-resolution is an ill-posed problem, prior information regarding the image to restore such as typical structures (features) in the image and the relationship between the structures is essential to the recovery of missing information. Inspired by this observation, we propose using this prior information as a regularizer for the latent fingerprint super-resolution problem.

Let *G* be a weighted graph with edge (a,b) that connects pixels *a, b* in image *I*, and let *G(a, b)* be the weight of the edge. The graph-total variation energy of image *I* according to *G* is defined as [107]:

$$\mathcal{J}_G(f) = \sum_a ||\nabla_a^G f||, \tag{5.3}$$

where $||.||$ is the euclidian norm over $\mathbb{R}^n$, f is a function, $\nabla_a^G f \in \mathbb{R}^n$ is a gradient vector defined for every pixel *a* in *I*,

$$\nabla_a^G f = (\sqrt{G(a,b)}((f(b) - f(a)))_b \tag{5.4}$$

Depth	**PSNR**	**SSIM**	**NIQE**	**Training** (mins.)	**Testing** (secs.)
5	18.153	0.8244	5.654	150.25	0.25
6	19.958	0.8685	4.324	166.15	0.32
7	20.635	0.8956	4.255	178.05	0.35
8	22.256	0.9057	4.295	183.65	0.38
9	24.382	0.9148	4.138	196.08	0.45
10	26.334	0.9228	4.037	210.54	0.49
11	29.734	0.9336	4.027	231.16	0.53
12	30.813	0.9466	4.025	242.55	0.65
13	31.968	0.9594	4.024	250.86	0.71
14	32.959	0.9618	4.019	256.45	0.75
15	**34.565**	**0.9695**	**4.009**	**320.08**	**0.85**
16	34.566	0.9696	4.007	375.21	1.52
17	34.567	0.9696	4.007	415.44	1.85
18	34.567	0.9697	4.006	500.05	1.96
19	34.568	0.9701	4.006	610.56	2.34
20	34.568	0.9702	4.005	713.45	3.15
21	34.569	0.9702	4.004	825.53	4.42

Table 5.1: Network depth vs. Computational Cost for scale factor 2. Each experiment involved 50 epochs, each 50 iterations for a total of 2,500 iterations. The same network parameters specified in section 5.3.2 were used in all the experiments. The row with the optimal depth is highlighted in bold.

We compute the weight between two pixels in the graph as the similarity between their 3x3 patch neighborhoods. For each pixels *i* in image *I*, its neighborhood patches are defined as 3x3 patches centered on *i*. We define the regularization functional as the weighted sum of square differences between all the pixel pairs in *I*. This definition may pose challenges in certain problem domains where initial images to use in determining the weights may not be available [146]. In our case, explicit dependence of the regularization penalty on predetermined weights is not an issue since latent fingerprints are available for computing the weights.

To obtain optimal pixel neighborhood window and patch sizes for computing the weights, we tried window size 3x3, 5x5, 7x7, 9x9 and 11x11, and patch size 5x5, 7x7, 9x9, 11x11 and 13x13. For each experiment, we recorded the computational cost of the regularizer, the peak signal-to-noise ratio (PSNR), and the structural similarity index (SSIM) of the super-resolved fingerprint. The results of the experiments are shown in Table 5.2. A plot of the results for the window sizes and the optimal patch size (7x7) based on the results in Table 5.2 are shown in Figure 5.4. It can be observed from the figure that using 3x3 as the pixel neighborhood window size and 7x7 as the patch size for the patch based regularization gave the best performance. The results are in line with the observation in [140] that the computational cost of the patch based regularization depends heavily on the patch size and neighborhood window size. Moreover, using large patches can prevent the algorithm from identifying small but relevant image features that can contribute to the quality of the output image.

We calculate the weight function w_{ij} on the low resolution latent fingerprint using patches:

$$w_{ij}(\tilde{x}) = \exp\left(- \frac{\|f_i(\tilde{x}) - f_j(\tilde{x})\|_h}{\delta^2} \right) \tag{5.5}$$

where $f_j(x)$ is a feature vector consisting of intensity values of all pixels in the patch

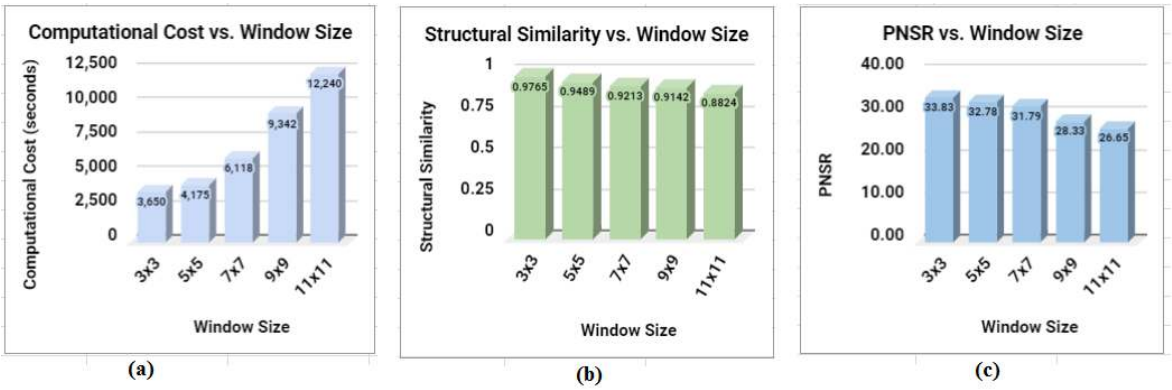


Figure 5.4: Neighborhood window size selection. The neighborhood window size is used in computing the weight between two pixels in an image patch (see equation (5.5). All plots are based on the specified window size and 7x7 image patch size. The plots show that using 3 x 3 as the pixel neighborhood window size and 7 x 7 as the patch size gave the best computational cost (lower is better), PSNR (higher is better) and SSI (higher is better). Experiments were performed with 2x upscaled input images.

centered at pixel *j*. The patch-based distance between pixel *j* and *k* is measured by

$$\|f_j(x) - f_k(x)\|_h = \sqrt{\sum_{p=1}^{n_p} h_p (x_{j_p} - x_{k_p})^2} \tag{5.6}$$

where j_p denotes the *pth* pixel in the neighborhood patch centered at *j* and k_p denotes the *pth* pixel in the neighborhood patch centered at pixel *k*. n_p is the total number of pixels in a patch. h_p is the normalized inverse spatial distance between pixel j_p and pixel *j* and is used as a positive weighting factor [140] with

$$\sum_{l=1}^{n_p} h_l = 1. \tag{5.7}$$

5.3 Experiments

In this section, we provide an evaluation of the performance of the proposed algorithm on three databases: one latent fingerprint database (NIST SD27) and two fingerprint databases

Window Size

Patch Size

	3 x 3	5 x 5	7 x 7	9 x 9	11 x 11
Cost (Secs.)					
3 x 3	-	-	-	-	-
5 x 5	3,513	-	-	-	-
7 x 7	3,650	4,175	-	-	-
9 x 9	3,718	4,820	6,118	-	-
11 x 11	3,868	4,938	6,325	9,342	-
13 x 13	4,033	4,112	6,844	9,849	12,240
SSIM					
3 x 3	-	-	-	-	-
5 x 5	0.9231	-	-	-	-
7 x 7	0.9765	0.9489	-	-	-
9 x 9	0.9632	0.9415	0.9213	-	-
11 x 11	0.9591	0.9322	0.9203	0.9142	-
13 x 13	0.9533	0.9218	0.9105	0.9038	0.8824
PSNR					
3 x 3	-	-	-	-	-
5 x 5	33.62	-	-	-	-
7 x 7	33.83	32.78	-	-	-
9 x 9	33.11	32.19	31.79	-	-
11 x 11	33.05	31.85	31.05	28.33	-
13 x 13	32.15	30.26	30.14	17.01	26.65

Table 5.2: Image patch size and neighborhood window size selection. The computational cost, SSIM and PSNR values for the various window size / patch size combinations are shown in the table. The 3 x 3 window size with 5 x 5 patch size has the best computational cost (lower is better), but using 3 x 3 as the neighborhood window size and 7 x 7 as the patch size, gave the best PSNR (higher is better) and SSIM (higher is better) performance. Experiments were performed with 2x upscaled input images. Based on the results of this experiment, 3 x 3 window size and 7 x 7 patch size were selected for patch based regularization of the proposed model.

(FVC2000 DB3_B and FVC2006 DB1_B). We describe performance evaluation metrics, the datasets used for training and testing, and the training parameters. Since there is no existing *latent fingerprint* SR method to compare with, we first provide benchmark results of the proposed model on NIST SD27 latent fingerprint database and then compare our method with the state-of-the-art fingerprint single image SR methods.

5.3.1 Performance Evaluation Metrics

We used the following image quality metrics to evaluate the performance the LIFSR network.

• **Peak Signal-to-Noise Ratio (PSNR):** PSNR is used as a quality measurement between an original image and reconstructed image. Larger PSNR values indicate better quality of the reconstructed image. PSNR is defined via the mean squared error (MSE):

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [K_1(i,j) - K_2(i,j)]^2, \tag{5.8}$$

where K_1 is the original image and K_2 is the reconstructed image. PSNR in decibel (dB) is defined as:

$$PSNR = 10 \log_{10} \left(\frac{V^2}{MSE} \right), \tag{5.9}$$

where V is the maximum possible pixel value of the image.

• **Structural Similarity Index (SSIM):** SSIM is used to assess the visual impact of luminance, contrast and structure of an image against a reference image.

$$SSIM(K_1, K_2) = [\ell(K_1, K_2).\varsigma(K_1, K_2).\mu(K_1, K_2)], \tag{5.10}$$

where K_1, K_2 are the test and reference images, respectively, the comparison functions ℓ is luminance, ς is contrast, and μ is structure. The closer to 1 the *SSIM* value is, the more similar the test and reference images are.

• **Naturalness Image Quality Evaluator (NIQE):** NIQE is a measure of perceptual image quality. Smaller NIQE scores indicate better perceptual quality. NIQE is a no-reference quality metric that is in agreement with a subjective human quality score [102].

5.3.2 Training Validation and Testing

Training Dataset

The training dataset was created from fingerprint images in the NIST SD4 database. This database consists of 2,000 fingerprint images. Each image is 512x512 pixels at 120 dpi. We randomly selected 1000 images from the database to create two datasets for training and validating the network. The training dataset consists of low-resolution images that have been upscaled using bicubic interpolation. The desired network responses (response dataset) used for validation are the residual images obtained by calculating the difference between the original images and their corresponding upscaled versions. The training data was fed to the network using a random patch extraction algorithm that extracted random corresponding patches from the training dataset (containing 1000 upscaled images) and response dataset (containing 1000 residual images).

Evaluation Datasets

The following three databases were used in evaluating the proposed model:

• **NIST SD27** [7]: This latent fingerprint database was acquired from the National Institute of Standards and Technology (NIST). It contains images of **258** latent crime scene fingerprints and their matching rolled tenprints. The 258 latent fingerprint images are at **500** dpi and consist of 88 Good, 85 Bad and 85 Ugly quality latent fingerprint images. The quality assigned to each image was based on the condition of the image in the location in which the minutia was positioned, and on how clearly identifiable the type of the minutia was in the image [7]. NIST has discontinued the distribution of SD27 database but has not announced a replacement.

• **FVC2000 DB3_B:** This fingerprint database is a publicly available database containing 80 finger-

print images at 500 dpi.

• **FVC2006 DB1_B** [34]: This is a low resolution publicly available fingerprint database containing 1800 fingerprint images at 250 dpi.

Training Parameters

We train the proposed model with 0.1 initial learning rate, 0.9 momentum and batch size of 64 and weight decay of 0.0001. Network initialization is done using a zero-mean Gaussian distribution with standard deviation $\sqrt{\frac{2}{L_n}}$ [68], where L_n in the number of layers in the network. We also initialize the bias to 0. Training was done for 50 epochs, with learning rate decreased by a factor of 10 after every 10 epochs. Our algorithms were implemented in Matlab R2018b running on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. The implementation relied on Matlab Deep Learning Toolbox. Training of the final network configuration took 10.5 hours to converge.

Figure 5.5 shows LAFISR SR results for sample images from NIST SD27 database. The residual image learnt by LAFISR as well as the ROIs before and after SR are shown in columns 3 and 4, respectively. The values obtained for the three performance evaluation metrics are also shown in the figure. Figure 5.6 shows sample feature maps at different layers of the LAFISR network. It can be observed that the filters in deeper layer of the network show more details.

5.3.3 Benchmark on NIST SD27

We performed experiments using NIST SD27 database to evaluate the effectiveness of LAFISR on latent fingerprint image processing.

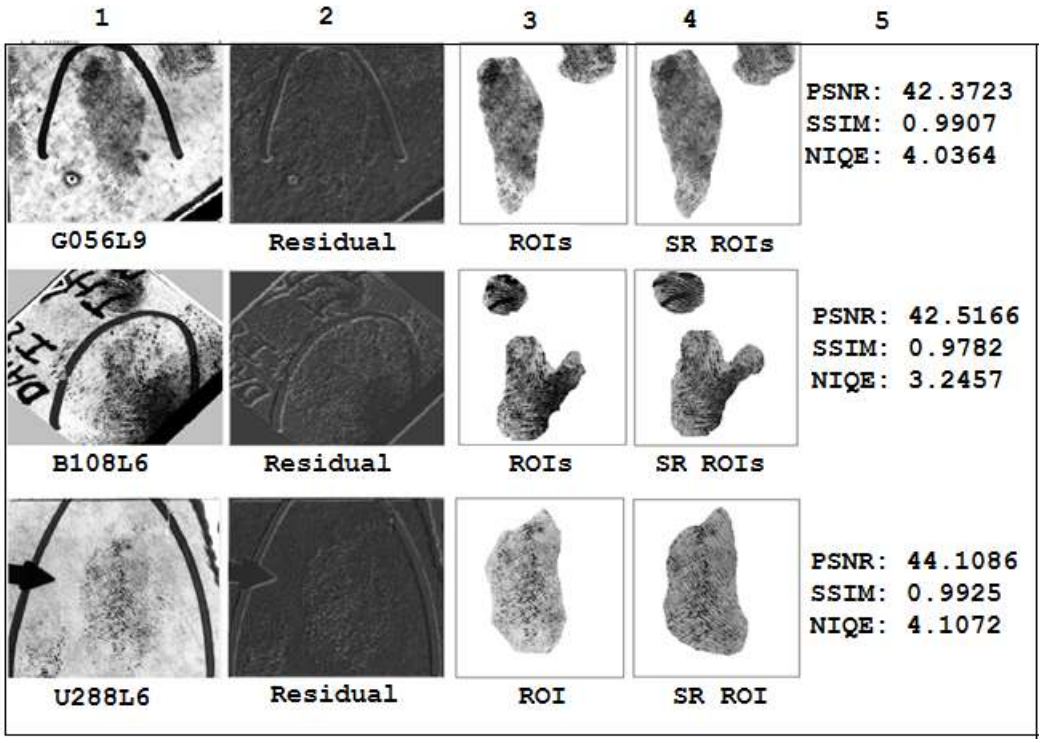


Figure 5.5: Results for sample NIST SD27 latent fingerprints. The original images are shown in column 1, the residual images learnt by the model are in column 2, the segmented ROIs before SR are in column 3, while the column 4 contains the segmented ROIs after SR. The values of the quality metrics obtained are shown in column 5.

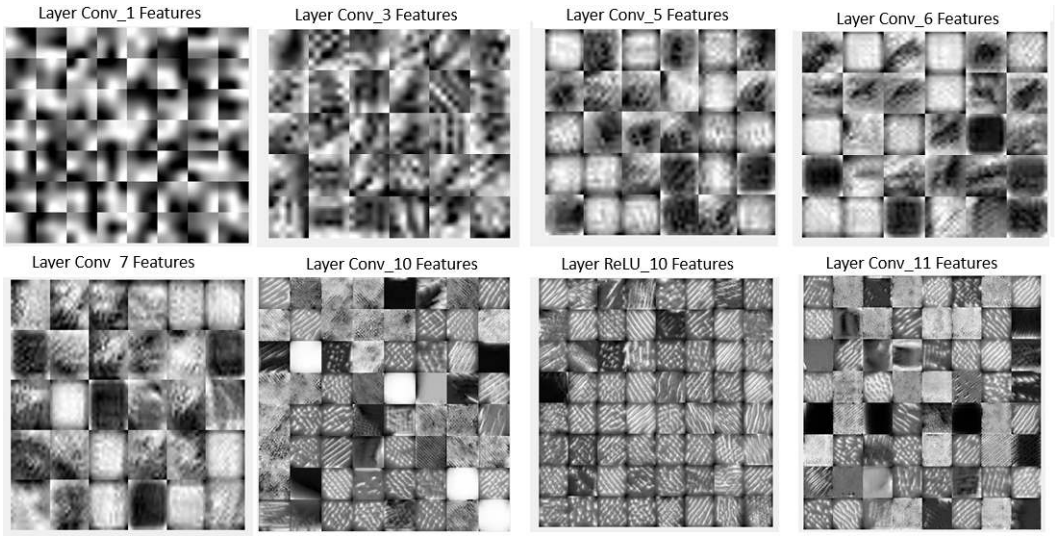


Figure 5.6: Sample feature maps at different layers (1, 3, 5, 6, 7, 10, 11) of the network. As can be seen from the figure, filters in deeper layers (10, 11) of the network show more details.

Latent Fingerprint Image Enhancement

To determine if LAFISR can be an effective pre-processing step for latent fingerprint feature extraction and matching, we evaluated its impact on improving the quality of latent fingerprint images. Using NIST MINDTCT [4] open source software, we assessed the quality of sample latent fingerprint region-of-interest (ROI) before and after SR with LAFISR by counting the number of minutiae detected in the before and after samples. Figure 5.7 shows the results for a sample latent fingerprint (NIST SD27 B106L8). For each of the MINDTCT minutiae quality settings shown, more features (minutiae) are detected after the latent fingerprint is enhanced via SR with LAFISR. This is due to improvement in the quality of the latent fingerprint after SR. Figure 5.8 is a plot of minutiae count against MINDTCT [4] minutiae quality settings for 20 randomly selected latent fingerprints from NIST SD27 database. It also shows that more features are detected by MINDTCT [4] after SR with LAFISR.

LAFISR vs. Bicubic Interpolation

We present quantitative comparison of bicubic interpolation and LAFISR methods in reconstructing high-resolution versions of latent fingerprints at three scale factors 2, 3 and 4. The three scale factors are widely used in SR comparisons. Table 5.3 shows the results using PSNR, SSIM, and NIQE quality metrics, for all 268 images in NIST SD27 database, as well as the three image categories (Good, Bad, Ugly). LAFISR outperforms Bicubic interpolation in all categories and all scale factors.

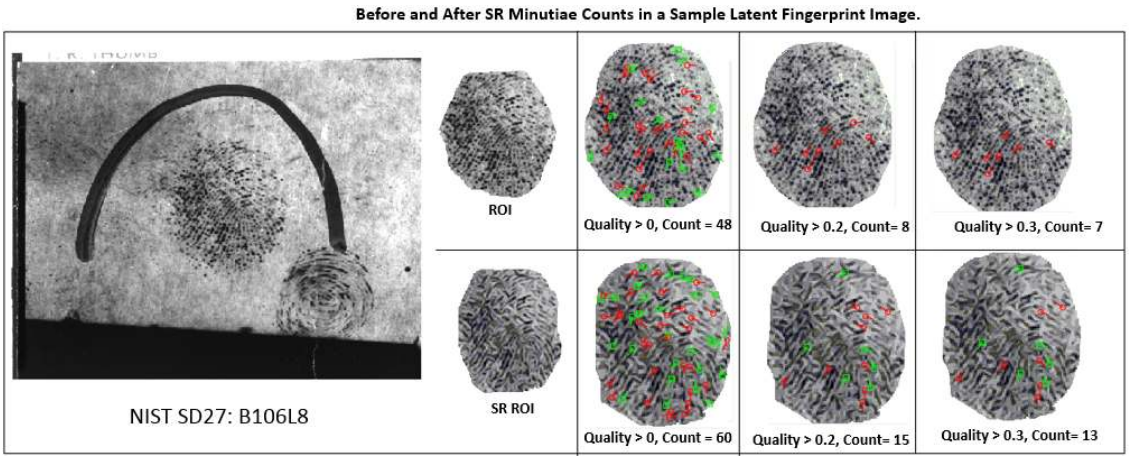


Figure 5.7: Before and after super-resolution (SR) minutiae quality and count for a sample NIST SD27 latent fingerprint. We used NIST MINDTCT [4] open source software to assess the quality and the number of minutiae in the region-of-interest (ROI) before and after SR. Top row shows the number of minutiae detected in the ROI of the input latent fingerprint for three minutiae quality settings ($> 0, > 0.2$ and > 0.3). The bottom row is for the ROI after SR. The results show that more features are detected after the latent fingerprint is super-resolved. This is due to the improvement in the quality of the latent fingerprint after SR. The images with minutiae indicated on them have been slightly enlarged for visual appeal. Bifurcations and Ridge endings are annotated in red and green color, respectively.

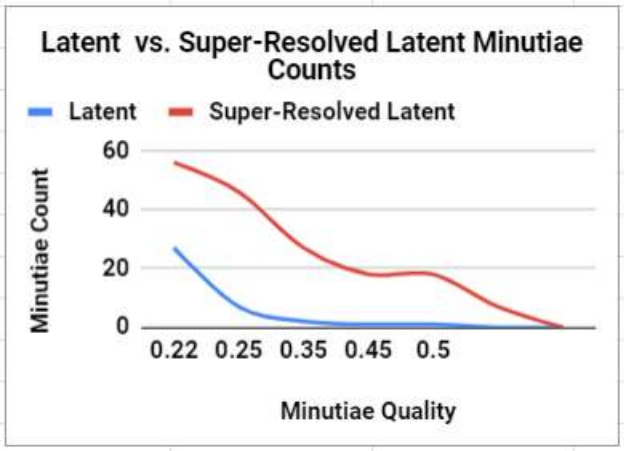


Figure 5.8: Latent fingerprint vs. super-resolved latent fingerprint minutiae counts. Please note that minutiae count determined using MINDTCT [4] may not be accurate. The point of the experiment is to show that with super-resolution, more features (minutiae) can be detected in the latent fingerprint.

Matching Performance

LAFISR can serve as an effective pre-processing step for latent fingerprint matching. In this section, we evaluate the matching performance to demonstrate the effectiveness of LAFISR in

Dataset	**Scale**	**Bicubic Interpolation**	**Proposed Method**
		PSNR / SSIM / NIQE	PSNR / SSIM / NIQE
NIST SD27 (Good: **Count=88**)	x2	35.8808 / 0.9683 / 4.9862	42.4743 / 0.9909 / 4.0395
	x3	29.1924 / 0.8866 / 5.7654	42.6355 / 0.9913 / 3.9810
	x4	27.4678 / 0.8148 / 6.2891	42.9221 / 0.9918 / 4.0098
NIST SD27 (Bad: **Count=85**)	x2	34.6682 / 0.9650 / 4.6033	42.6198 / 0.9762 / 3.2948
	x3	28.2405 / 0.8788 / 5.4004	42.7423 / 0.9764 / 3.2814
	x4	26.5369 / 0.8100 / 5.5461	42.9108 / 0.9763 / 3.2868
NIST SD27 (Ugly: **Count=85**)	x2	34.5318 / 0.9277 / 5.0184	44.1086 / 0.9946 / 4.1037
	x3	28.7839 / 0.8112 / 6.0570	44.2278 / 0.9950 / 4.1150
	x4	27.6857 / 0.7365 / 5.9128	44.3969 / 0.9953 / 4.0884
NIST SD27 (Good, Bad, Ugly: **Count=258**)	x2	35.0269 / 0.9537 / 4.8693	43.0676 / 0.9872 / 3.8127
	x3	28.7389 / 0.8589 / 5.7409	43.2019 / 0.9876 / 3.7925
	x4	27.2302 / 0.7871 / 5.9160	43.4099 / 0.9878 / 3.7950

Table 5.3: Average PSNR/SSIM/NIQE for scale factors 2, 3 and 4 on NIST SD27 database Good, Bad and Ugly image categories. LAFISR outperforms Bicubic interpolation on all three performance measures. For PSNR and SSIM, bigger is better, while smaller is better for NIQE.

improving matching results. We compare the matching results obtained using the dataset containing original NIST SD27 latent fingerprints and a dataset containing super-resolved versions of NIST SD27 latent fingerprints. For this experiment, we used a reference dataset of 5,257 images consisting of 257 rolled fingerprints in NIST SD27 database and 5,000 synthetic fingerprints generated with SFinGe (Synthetic Fingerprint Generator) [10]. Synthetic fingerprints have been shown to be very useful for training and testing purposes, and have been used for technology evaluations [2]. Figure 5.9 shows the results of the evaluation. There is slight improvement in matching performance when the super-resolved latent fingerprints are used. The improvement increases as we move from Good quality latent to Ugly quality latents. The results highlight the promise of LAFISR in pre-processing latent fingerprints, especially for low quality ones.

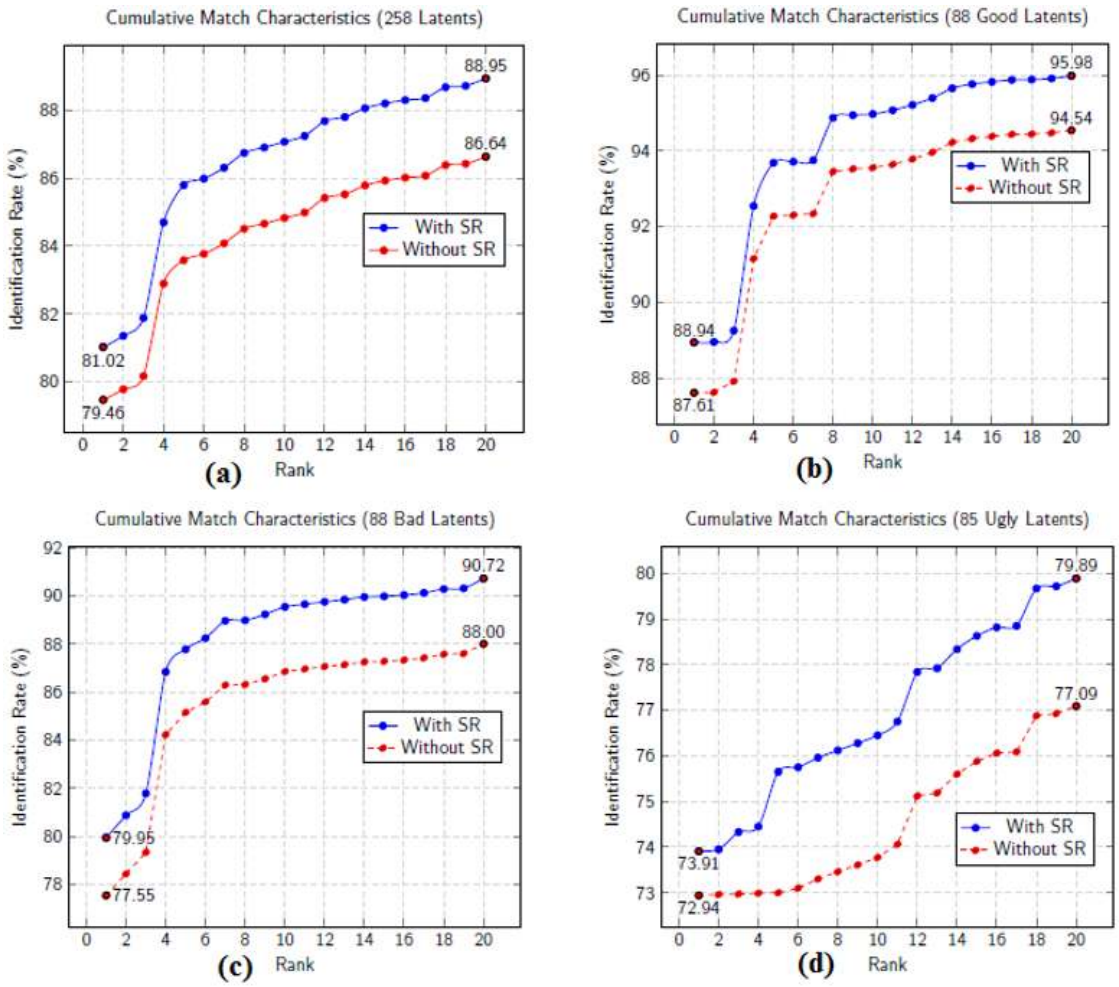


Figure 5.9: NIST SD27: CMC plots of the proposed approach in matching all, as well as the three categories of latent fingerprints in NIST SD 27 database against a reference database of 5,257 rolled fingerprints. (a) All (258), (b) Good (88), (c) Bad (85), and (d) Ugly (85) latent fingerprints. The plots show that matching performance is improved when the super-resolved latent fingerprints are used. The amount of improvement increases as we move from Good quality latent to Ugly quality latents.

Quality Prediction

We also compare the quality predictions of the model in [54] using the super-resolved latent fingerprints as input with the VID, VEO, and NV value determination by latent examiners [70], the quality value predictions by Expert Crowd [43] and the quality predictions of the model in [54] using the original latent fingerprints as input. As stated in [54], NIST SD27 database is the

	VID	**VEO**	**NV**
Latent Examiners [70]	155/210	**11**/41	0/7
Chugh et al. (Expert Crowd) [43]	161/210	5/41	0/7
Ezeobiejesi et al. [54]	164/210	4/41	0/7
LAFISR	**166**/210	7/41	0/7

Table 5.4: NIST SD27 latent fingerprints retrieved at Rank-1 using a state-of-the-art latent AFIS. The results show that using LAFISR super-resolved latent fingerprints leads to improvement in predicting latent AFIS performance for VID (166/201). Latent examiners obtained better VEO than LAFISR 11/41 against 7/41.

only latent database with available latent value determinations by latent examiners. According to [70], there are 210 VID, 41 VEO, and 7 NV latents in NIST SD27. A total of 166 latents (155 VID , and 11 VEO) out of the 256 latents in NIST SD27 are retrieved at Rank-1 using the state-of-the-art latent AFIS [43]. We follow the protocol used in [43] and the steps outlined in [54]. Table 5.4 shows a comparison of the number of latents retrieved at rank-1 using value determination by latent examiners [70], the predicted latent value from [43], the predicted latent quality from [54], and the predicted latent quality using LAFISR super-resolved latent fingerprint. This performance comparison was done using a reference dataset containing 5,257 fingerprints created from 5,000 synthetic fingerprints generated with SFinGe, and the 257 rolled images in NIST SD27 database. With respect to predicting latent AFIS performance, the quality prediction using LAFISR super-resolved latent fingerprints is better than the value determination latent examiners and value prediction by both Expert Crowd and the model in [54].

5.3.4 Comparison with Other Methods

Finally, we present the qualitative and quantitative performance comparison our model with the sate-of-the-art fingerprint image super-resolution models. To the best of our knowledge, this is the first work that uses super-resolution for *latent fingerprint* image enhancement. Since there is no existing SR work on latent fingerprints, we present both quantitative and visual comparison of the performance of our model with existing fingerprint super-resolution papers [126] and [27]. We present comparison with [126, 27] since the authors used a publicly available fingerprint databases (FCV2000, and FCV2006) for performance evaluation. For quantitative comparison we use two quality measures namely PSNR and SSIM.

Table 5.5 shows the PSNR and SSIM values for sample images from FVC2000 database for the method used in [126] and the proposed method. The PSNR and SSIM for [126] are from published results. For all scale values, the proposed method achieves higher PSNR and SSIM (higher is better). The results show that the quality of the SR is better with our proposed method. The results also show that the performance of our method is almost stable across the scales factors. This is because LAFISR is trained with scale augmentation [81] and is, therefore, capable of performing SR at multiple scale factors without appreciable performance degradation.

Figures 5.10 show visual comparison of the SR images reconstructed by SR methods in [126, 27] and the proposed method for scale factors 2, 3 and 4. The results show that our proposed method performs better than existing SR fingerprint enhancement algorithms both in terms of high resolution details and minimization of artifacts. Also, the SR images from our algorithm are visually closer to the respective input images.

Image	**Scale**	**Singh et al. [126]** PSNR / SSIM	**Proposed Method** PSNR / SSIM
FVC2000 DB3_B 101_1			
	x2	- / -	42.2553/0.9976
	x3	22.6510/0.7831	**43.3887 / 0.9981**
	x4	22.1582/0.7123	**44.2847 / 0.9986**
FVC2000 DB3_B 105_1			
	x2	- / -	44.0220 / 0.9974
	x3	19.5281/0.7093	**44.3530 / 0.9978**
	x4	19.3793/0.6500	**44.7160 / 0.9980**
Mean (FVC2000 DB3_B: 50 Images)			
	x2	- / -	43.1386 / 0.9975
	x3	22.1869/0.7698	**43.8708 / 0.9982**
	x4	21.9754/0.7021	**44.5003 / 0.9984**
Mean (FVC2006 DB1_B: 50 Images)			
	x2	- / -	42.8115 / 0.9875
	x3	- / -	42.9758 / 0.9965
	x4	- / -	43.3417 / 0.9893

Table 5.5: PSNR values (in dB) and SSIM for the SR output images by Singh et al. [126], and our method at scale factors 2, 3 and 4. Our Method outperforms that of Singh et al. (higher is better). No results for scale factor 2 are provided in [126]. The performance of our model is also stable across scale factors because it is trained with scale augmentation making it capable of performing SR at multiple scales without appreciable performance degradation.

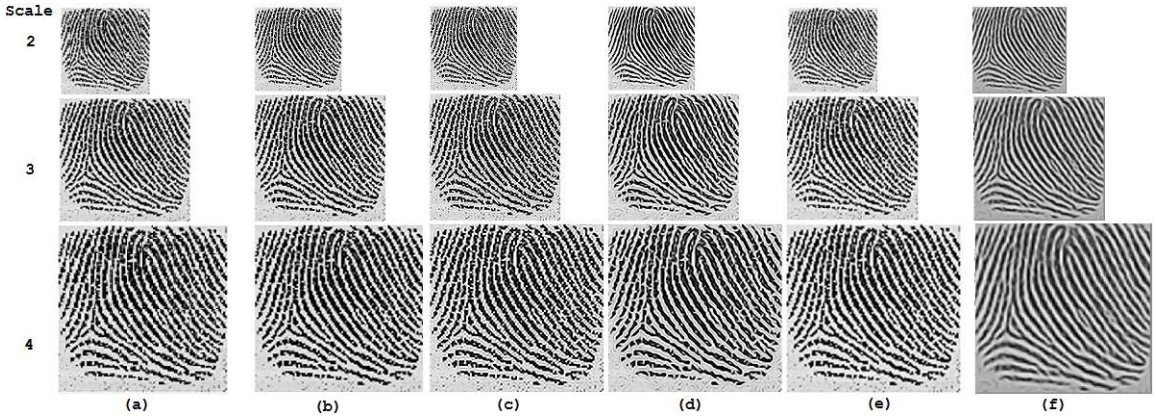


Figure 5.10: A visual comparison of the SR results of different methods at scale factors 2, 3 and 4 in rows 1, 2 and 3, respectively, for sample real low resolution fingerprint 110_2 from FVC2006 DB1_B database [34]. The original image at the various scale factors is shown in column (a). Columns (b) through (f) show the SR results from various methods: (b) Bicubic interpolation, (c) Singh et al. [126], (d) Bian et al. [27], (e) Our method (LAFISR), and (f) enhanced output from LAFISR. The results show that LAFISR produces significantly sharper images than the other algorithms. The images in columns (c) and (d) are from the published results [126, 27].

5.3.5 Patch Regularized vs. Non Patch Regularized LAFISR

We performed experiments to investigate the impact of patch based regularization on the performance of the proposed model. We trained two models, one with patch based regularization and the other without it. We tested the two models using five images selected from NIST SD27 database. Table 5.6 shows the PSNR and SSIM values obtained for the sample latent fingerprints from NIST SD27 database for the two models. The results show that better performance is achieved when the model is trained with patch based regularization. Figure 5.12 shows a visual comparison of the SR results obtained from the two models for a sample latent fingerprint from NIST SD27 database. Sharper super-resolved latent is obtained using patch regularized LAFISR.

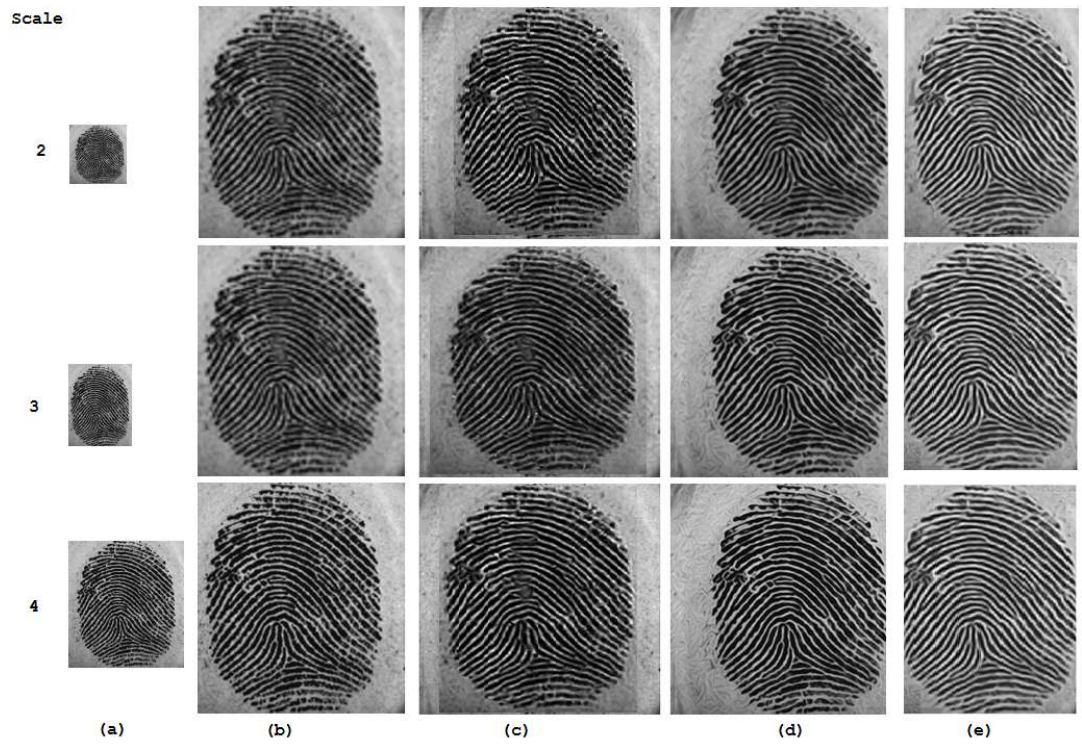


Figure 5.11: A visual comparison of the SR results of different methods at scale factors 2, 3 and 4 in rows 1, 2 and 3, respectively, for sample synthetic low resolution fingerprint 101_1 from FVC2000 DB3_B database. The original image at various scale factors is shown in column (a). Columns (b) through (e) show the SR results from various methods: (b) Bicubic interpolation, (c) Singh et al. [126], (d) Bian et al. [27], and (e) Our method (LAFISR). It can be seen that LAFISR produces significantly sharper images than the other algorithms. The images in columns (c) and (d) are from the published results [126, 27].

LAFISR Computational Cost

The training of the final configuration of LAFISR (15 convolutional layers, 15 ReLU layers, 1 regression layer) took 10.5 hours to converge. The average processing time for a single image SR using the trained model is 1200ms. Given that reconstruction accuracy is of utmost importance for improving matching results, the reconstruction accuracy of our proposed model is worth the computational cost. Computing the regularization penalty takes 40% of the training time. The regularization penalty depends on predetermined weights between pixels in the image patches

Image	LAFISR PSNR / SSIM	LAFISR + Patch Reg. PSNR / SSIM
G056L9	32.2267 / 0.9521	45.8854 / 0.9913
G053L6	32.1543 / 0.9346	44.9587 / 0.9887
G055L3	33.2882 / 0.9526	43.9254 / 0.9783
B106L8	32.6543 / 0.9352	43.4436 / 0.9794
U288L6	29.4521 / 0.9416	42.1124 / 0.9798

Table 5.6: PSNR and SSIM values for sample latent fingerprints from NIST SD27 database obtained with patch based regularization (LAFISR + Patch Reg.), and without patch based regularization (LAFISR). Better results were obtained with LAFISR + Patch Reg.

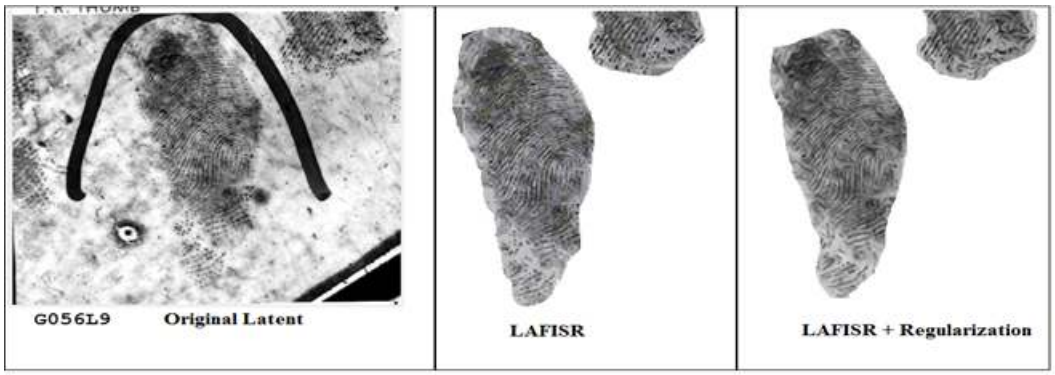


Figure 5.12: Reconstruction results on sample NIST SD27 latent fingerprint. The original image is shown on the left, the super-resolved version using LAFISR is shown in the middle, while the super-resolved one with LAFISR + Patch regularization is shown on the right. The figure shows that sharper super-resolved latent is obtained using patch regularized LAFISR.

used to train the model. This weight between two pixels is a measure of similarity between the 3x3 patch neighborhoods centered on the specified pixels. The training (re-training) time can be reduced by pre-computing the regularization penalty weights and doing a table lookup during training. This strategy will be considered in future work.

5.4 Summary

This chapter presented a framework (LAFISR) for latent fingerprint image super-resolution using deep neural networks. The proposed algorithm used graph-total variation energy of latent fingerprints as priors to regularize the ill-posed super-resolution problem. The regularizer penalized the model towards learning optimal weights, leading to high quality image reconstruction results. We evaluated the quality of the super-resolved latent fingerprint images by comparing the matching results obtained using a dataset containing original NIST SD27 latent fingerprints and a dataset containing super-resolved versions of the same images. The results showed improved matching performance using LAFISR pre-processed latent fingerprints, especially for low quality latents. Qualitative and quantitative performance comparison of our model with other fingerprint image super-resolution models highlighted the superior performance of LAFISR. To the best of our knowledge, this is the first work that used super-resolution for latent fingerprint image enhancement.

Chapter 6

Conclusions and Future Work

This dissertation presented deep learning models and algorithms developed in the context of machine learning for automatic latent fingerprint image quality assessment, quality improvement, segmentation and matching. To the best of our knowledge, this is the first deep learning based end-to-end automatic framework that addresses the problems inherent in latent fingerprint quality assessment, quality improvement, segmentation and matching. The framework includes a unified frequency domain based model for latent fingerprint matching using image patches, a novel latent fingerprint super-resolution model that uses a graph-total variation energy of latent fingerprints as a non-local regularizer for learning optimal weights for high quality image reconstruction, as well as techniques that help speed-up convergence of a deep neural network and achieve a better estimation of the relation between a latent fingerprint image patch and its target class.

Latent fingerprint comparison is increasingly relied upon by law enforcement to solve crime, and prosecute offenders. The increasing use of this service places new strains on the limited resources of the forensic science delivery system. Currently, latent examiners manually mark the

region of interest (ROI) in latent fingerprints and use features manually identified in the ROI to search large databases of reference full fingerprints to identify a small number of potential matches for subsequent manual examination. Given the large size of law enforcement databases containing rolled and plain fingerprints, it is very desirable to perform latent fingerprint processing in a fully automated way. The framework and models presented in this dissertation will eliminate the manual processing of latent fingerprint images and lead to significant improvement in the matching accuracy of latent fingerprints. The automatic feature extraction performed with our deep learning model will improve the repeatability and reproducibility of latent fingerprint identification and ultimately reduce the manual and tedious work done by latent examiners.

6.0.1 Future Work

There are some interesting extensions of the methods presented in this dissertation. For quality assessment, our future work will involve using NIST Finger Image Quality (NFIQ 2.0) as a baseline for mapping latent fingerprint quality assessment to recognition performance. We intend to explore the performance of the patch based latent fingerprint matching presented in chapter 4 on a fingerprint database with mixed images resolutions. Future extensions of the work presented in chapter 5 will include learning the regularization priors of the weights of the neural networks and designing learned filters such that input images can be directly mapped to high resolution versions without the interpolation stage.

Bibliography

[1] In *https://goo.gl/5wmG3H*.

[2] FVC2004. *http://bias.csr.unibo.it/fvc2004*.

[3] Integrated Pattern Recognition and Biometrics Lab, West Virginia University. *http://www.csee.wvu.edu/*.

[4] MINDTCT. *Fingerprint Minutiae Detector.* *http://www.nist.gov/services-resources/software/nist-biometric-image-software-nbis*.

[5] NIFQ 2.0. *https://www.nist.gov/sites/default/files/documents /2018/11/29/nfiq2_report.pdf*.

[6] NIST SD14. *https://www.nist.gov/srd/nist-special-database-14*.

[7] NIST Special Database 27. *Fingerprint Minutiae from Latent and Matching Ten-print Images.* *http://www.nist.gov/srd/nistsd27.htm*.

[8] NIST Special Database 4. *NIST 8-Bit Gray Scale Images of Fingerprint Image Groups (FIGS)*.

[9] Regionprops. *https://www.mathworks.com/help/images/ref/regionprops.html*.

[10] SFINGE. *http://biolab.csr.unibo.it/sfinge.html*.

[11] Verifinger SDK 2018. *Neurotechnology Inc.,* *http://www.neurotechnology.com/verifinger.html*.

[12] E. Ahmed, M. Jones, and T. K. Marks. An improved deep learning architecture for person re-identification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916, June 2015.

[13] O.S. Al-Kadi and D. Watson. Texture analysis of aggressive and nonaggressive lung tumor CE CT images. *Biomedical Engineering, IEEE Transactions on*, 55(7):1822–1830, July 2008.

[14] F. Alonso-Fernandez, J. Fierrez, J. Ortega-Garcia, J. Gonzalez-Rodriguez, H. Fronthaler, K. Kollreider, and J. Bigun. A comparative study of fingerprint image-quality estimation methods. *IEEE Transactions on Information Forensics and Security*, 2(4):734–743, Dec 2007.

[15] Le An and Bir Bhanu. Face image super-resolution using 2D CCA. *Signal Processing*, 103:184 – 194, 2014.

[16] A.Napolitano, S.Ungania, and V.Cannata. Fractal dimension estimation for biomedical images. In *MATLAB - A Fundamental tool for scientific computing and Engineering Applications*, volume 3, pages 161–772, 2012.

[17] I. Arel, D. C. Rose, and T. P. Karnowski. Deep machine learning - a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, 5(4):13–18, Nov 2010.

[18] S. S. Arora, E. Liu, K. Cao, and A. K. Jain. Latent fingerprint matching: Performance gain via feedback from exemplar prints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2452–2465, Dec 2014.

[19] Irfan Arshad, Gulistan Raja, and AhmadKhalil Khan. Latent fingerprints segmentation: Feasibility of using clustering-based automated approach. *Arabian Journal for Science and Engineering*, 39(11):7933–7944, 2014.

[20] Irfan Arshad, Gulistan Raja, and AhmadKhalil Khan. Latent fingerprints segmentation: Feasibility of using clustering-based automated approach. *Arabian Journal for Science and Engineering*, 39(11):7933–7944, 2014.

[21] MN Barros Filho and FJA Sobreira. Accuracy of lacunarity algorithms in texture classification of high spatial resolution images from urban areas. In *XXI congress of international society of photogrammetry and remote sensing*, 2008.

[22] J. S. Bartunek, M. Nilsson, J. Nordberg, and I. Claesson. Adaptive fingerprint binarization by frequency domain analysis. In *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, pages 598–602, Oct 2006.

[23] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 153–160, Cambridge, MA, USA, 2006. MIT Press.

[24] Matt A. Bernstein, Sean B. Fain, and Stephen J. Riederer. Effect of windowing and zero-filled reconstruction of MRI data on spatial resolution and acquisition strategy. *Journal of Magnetic Resonance Imaging*, 14(3):270–280, 2001.

[25] B. Bhanu and Xuejun Tan. Learned templates for feature extraction in fingerprint images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR*, volume 2, pages II–591–II–596 vol.2, 2001.

[26] Bir Bhanu and Yingqiang Lin. Genetic algorithm based feature selection for target detection in SAR images. *Image and Vision Computing*, 21(7):591 – 608, 2003.

[27] Weixin Bian, Shifei Ding, and Yu Xue. Fingerprint image super resolution using sparse representation with ridge pattern prior by classification coupled dictionaries. *IET Biometrics*, 6:342–350(8), September 2017.

[28] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, Aug 2007.

[29] K. Cao, T. Chugh, Jiayu Zhou, E. Tabassi, and A. K. Jain. Automatic latent value determination. In *International Conference on Biometrics (ICB)*, pages 1–8, June 2016.

[30] K. Cao and A. K. Jain. Latent orientation field estimation via convolutional neural network. In *2015 International Conference on Biometrics (ICB)*, pages 349–356, May 2015.

[31] 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*, 36(9):1847–1859, Sept 2014.

[32] 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*, 36(9):1847–1859, Sept 2014.

[33] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain. Performance evaluation of fingerprint verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):3–18, Jan 2006.

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

[35] Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *AISTATS*, volume 10, pages 33–40. Citeseer, 2005.

[36] H Chang. Super-resolution through neighbor embedding. *CVPR*, pages 1–8, 01 2014.

[37] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[38] Yi Chen, Sarat C. Dass, and Anil K. Jain. Fingerprint quality indices for predicting authentication performance. In *Proceedings of the 5th International Conference on Audio- and Video-Based Biometric Person Authentication*, AVBPA'05, pages 160–170, Berlin, Heidelberg, 2005. Springer-Verlag.

[39] H. Choi, M. Boaventura, I.A.G. Boaventura, and A.K. Jain. Automatic segmentation of latent fingerprints. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 303–310, Sept 2012.

[40] Huang Chongfu. Deriving samples from incomplete data. In *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)*, volume 1, pages 645–650 vol.1, May 1998.

[41] Jinghui Chu, Jiaqi Zhang, Wei Lu, and Xiangdong Huang. A novel multiconnected convolutional network for super-resolution. *IEEE Signal Processing Letters*, 25(7):946–950, Jul 2018.

[42] T. Chugh, S. S. Arora, A. K. Jain, and N. G. Paulter. Benchmarking fingerprint minutiae extractors. In *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–8, Sept 2017.

[43] T. Chugh, K. Cao, J. Zhou, E. Tabassi, and A. K. Jain. Latent fingerprint value prediction: Crowd-based learning. *IEEE Transactions on Information Forensics and Security*, 13(1):20–34, Jan 2018.

[44] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, June 2012.

[45] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[46] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

[47] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016.

[48] Claude E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979.

[49] V. N. Dvornychenko. Evaluation of fusion methods for latent fingerprint matchers. In *2012 5th IAPR International Conference on Biometrics (ICB)*, pages 182–188, March 2012.

[50] V. N. Dvornychenko and M. D. Garris. Summary of nist latent fingerprint testing workshop. NISTIR 7377, 2006, 2006.

[51] Meng Joo Er, W. Chen, and Shiqian Wu. High-speed face recognition based on discrete cosine transform and RBF neural networks. *IEEE Transactions on Neural Networks*, 16(3):679–691, May 2005.

[52] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.

[53] J. Ezeobiejesi and B. Bhanu. Latent fingerprint image segmentation using fractal dimension features and weighted extreme learning machine ensemble. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[54] J. Ezeobiejesi and B. Bhanu. Latent fingerprint image quality assessment using deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[55] Jude Ezeobiejesi and Bir Bhanu. Latent fingerprint image segmentation using deep neural network. In *Deep Learning for Biometrics*, pages 83–107, 2017.

[56] Jude Ezeobiejesi and Bir Bhanu. Patch based latent fingerprint matching using deep learning. In *ICIP*, 2018.

[57] Rong-En Fan and Chih-Jen Lin. A study on threshold selection for multi-label classification. *Department of Computer Science, National Taiwan University*, pages 1–23, 2007.

[58] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In *IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692, Sept 2016.

[59] J. Feng and A. K. Jain. Filtering large fingerprint database for latent matching, Dec 2008.

[60] Jianjiang Feng, Soweon Yoon, and Anil K. Jain. Latent fingerprint matching: Fusion of rolled and plain fingerprints. In Massimo Tistarelli and Mark S. Nixon, editors, *Advances in Biometrics*, pages 695–704, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[61] Anil Gavade and Prasana Sane. Super resolution image reconstruction by using bicubic interpolation. 10 2013.

[62] F. A. Georgescu, C. Vaduva, D. Raducanu, and M. Datcu. Feature extraction for patch-based classification of multispectral earth observation images. *IEEE Geoscience and Remote Sensing Letters*, 13(6):865–869, June 2016.

[63] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *IEEE 12th International Conference on Computer Vision*, pages 2335–2342, Sept 2009.

[64] Mohammad Haghighat, Saman Zonouz, and Mohamed Abdel-Mottaleb. Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21):7905 – 7916, 2015.

[65] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, June 2015.

[66] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973.

[67] R.M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. page 610–621. IEEE Trans. on Systems, Man and Cybernetics, 1973.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[69] R.A. Hicklin. Latent fingerprint quality: a survey of examiners. 61:385–418, 01 2011.

[70] R.A. Hicklin, JoAnn Buscaglia, Maria Antonia Roberts, Stephen B. Meagher, William Fellner, Mark J. Burge, Matthew Monaco, David Vera, Larry R. Pantzer, Calvin C. Yeung, and Ted N. Unnikumaran. Latent fingerprint quality: a survey of examiners. *Journal of Forensic Identification*, 61:385–419, 07 2011.

[71] G. Hinton. A practical guide to training restricted boltzmann machines, version 1. 2010.

[72] Chongfu Huang and Claudio Moraga. A diffusion-neural-network for learning from small samples. *International Journal of Approximate Reasoning*, 35(2):137 – 161, 2004.

[73] G.B. Huang, Q.Y Zhu, and C.K Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489 – 501, 2006.

[74] M. Indovina, R. A. Hicklin, and G. I. Kiebuzinski. Elft efs: Evaluation of latent fingerprint technologies: Extended feature sets [evaluation no. 1]. NISTIR 7775,2011, 2011.

[75] A. K. Jain and J. Feng. Latent fingerprint matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):88–100, Jan 2011.

[76] Tsai-Yang Jea and Venu Govindaraju. A minutia-based partial fingerprint recognition system. *Pattern Recognition*, 38(10):1672 – 1684, 2005.

[77] J.Ngiam, P.Koh, Z. Chen, S. Bhaskar, and A. Y. Ng. Sparse filtering.

[78] S. Karimi-Ashtiani and C.-C.J. Kuo. A robust technique for latent fingerprint image segmentation and enhancement. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1492–1495, Oct 2008.

[79] D.H. Kaye, T. Busey, M. Gische, G. LaPorte, C. Aitken, S.M. Ballou, L. Butt ..., and K. Wertheim. Latent print examination and human factors: Improving the practice through a systems approach. *NIST Interagency/Internal Report (NISTIR) - 7842*, Jan 2012.

[80] J.M. Keller, S. Chen, and R.M Crownover. Texture description and segmentation through fractal geometry. *Computer Vision, Graphics, and Image Processing*, 45(2):150 – 166, 1989.

[81] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, June 2016.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[83] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[84] Matthew Lai. Deep learning for medical image segmentation. *arXiv preprint arXiv:1505.02000*, 2015.

[85] A. D. K. T. Lam and Q. Li. Fractal analysis and multifractal spectra for the images. In *Computer Communication Control and Automation (3CA), 2010 International Symposium on*, volume 2, pages 530–533, May 2010.

[86] Pascal Lamblin and Yoshua Bengio. Important gains from supervised fine-tuning of deep architectures on large labeled sets. In *NIPS* 2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

[87] Yuan Lan, Yeng Chai Soh, and Guang-Bin Huang. Ensemble of online sequential extreme learning machine. *Neurocomputing*, 72(13–15):3391 – 3395, 2009. Hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007).

[88] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[89] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[90] Jian Li, Jianjiang Feng, and C.-C. Jay Kuo. Deep convolutional neural network for latent fingerprint enhancement. *Signal Processing: Image Communication*, 60:52 – 63, 2018.

[91] Shutao Li, James T. Kwok, and Yaonan Wang. Combination of images with diverse focuses using the spatial frequency. *Information Fusion*, 2(3):169 – 176, 2001.

[92] Eyung Lim, Xudong Jiang, and Weiyun Yau. Fingerprint quality and validity analysis. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–469–I–472 vol.1, 2002.

[93] Yunjie Liu, Evan Racah, Prabhat, Joaquin Correa, Amir Khosrowshahi, David Lavers, Kenneth Kunkel, Michael Wehner, and William D. Collins. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *CoRR*, abs/1605.01156, 2016.

[94] Z. Liu, W. Siu, and J. Huang. Image super-resolution via weighted random forest. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 1019–1023, March 2017.

[95] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[96] Zoltán Makó. Approximation with diffusion-neural-network. In *6th International Symposium of Hungarian Researchers on Computational Intelligence*, pages 18–19, 2005.

[97] D. Maltoni, D. Maio, A. K. Jain, , and S. Prabhakar. Lfiq: Latent fingerprint image quality. In *Handbook of Fingerprint Recognition*. Springer Publishing Company, Incorporated, 2009.

[98] B.B. Mandelbrot. *The Fractal Geometry of Nature*. Einaudi paperbacks. Henry Holt and Company, 1983.

[99] M. A. Medina-Pérez, A. M. Moreno, M. A. Ferrer Ballester, M. García-Borroto, O. Loyola-González, and L. Altamirano-Robles. Latent fingerprint identification using deformable minutiae clustering. *Neurocomputing*, 175:851 – 865, 2016.

[100] G. R. Mettam and L. B. Adams. How to prepare an electronic version of your article. In B. S. Jones and R. Z. Smith, editors, *Introduction to the Electronic Age*, pages 281–304. E-Publishing Inc., New York, NY, 1999.

[101] Bilal Mirza, Zhiping Lin, and Kar-Ann Toh. Weighted online sequential extreme learning machine for class imbalance learning. *Neural Processing Letters*, 38(3):465–486, 2013.

[102] A. Mittal, R. Soundararajan, and A. C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, March 2013.

[103] Andrew Ng, Jiquan Ngiam, and Chuan Yu Foo. http://ufldl.stanford.edu/tutorial/supervised/optimizationstocha *UFLDL Tutorial*.

[104] Dinh-Luan Nguyen, Kai Cao, and Anil K. Jain. Automatic cropping fingermarks: Latent fingerprint segmentation. *CoRR*, abs/1804.09650, 2018.

[105] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, 20(3):21–36, May 2003.

[106] A. A. Paulino, J. Feng, and A. K. Jain. Latent fingerprint matching using descriptor-based hough transform. *IEEE Transactions on Information Forensics and Security*, 8(1):31–45, Jan 2013.

[107] Gabriel Peyré, Sébastien Bougleux, and Laurent Cohen. Non-local regularization of inverse problems. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 57–68, Berlin, Heidelberg, 2008. Springer-Verlag.

[108] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[109] Yaniv Romano, John Isidoro, and Peyman Milanfar. Raisr: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, 3(1):110–125, Mar 2017.

[110] A. Rosenfeld. Editor: Digital picture analysis. 1976.

[111] P. Ruangsakul, V. Areekul, K. Phromsuthirak, and A. Rungchokanun. Latent fingerprints segmentation based on rearranged fourier subbands. In *International Conference on Biometrics (ICB)*, pages 371–378, May 2015.

[112] P. Ruangsakul, V. Areekul, K. Phromsuthirak, and A. Rungchokanun. Latent fingerprints segmentation based on rearranged fourier subbands. In *International Conference on Biometrics (ICB)*, pages 371–378, May 2015.

[113] Muhammad Sajjad, Irfan Mehmood, and Sung Wook Baik. Image super-resolution using sparse coding over redundant dictionary based on effective image representations. *Journal of Visual Communication and Image Representation*, 26:50 – 65, 2015.

[114] Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[115] A. Sankaran, P. Pandey, M. Vatsa, and R. Singh. On latent fingerprint minutiae extraction using stacked denoising sparse autoencoders. In *IEEE International Joint Conference on Biometrics*, pages 1–7, Sept 2014.

[116] A. Sankaran, M. Vatsa, and R. Singh. Hierarchical fusion for matching simultaneous latent fingerprint. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 377–382, Sept 2012.

[117] A. Sankaran, M. Vatsa, and R. Singh. Automated clarity and quality assessment for latent fingerprints. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–6, Sept 2013.

[118] A. Sankaran, M. Vatsa, and R. Singh. Latent fingerprint matching: A survey. *IEEE Access*, 2:982–1004, 2014.

[119] N. Sarkar and B. B. Chaudhuri. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):115–120, Jan 1994.

[120] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. APAC: augmented pattern classification with neural networks. *CoRR*, abs/1505.03229, 2015.

[121] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3791–3799, June 2015.

[122] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528, June 2006.

[123] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2809–2813, July 2011.

[124] N.J. Short, M.S. Hsiao, A.L. Abbott, and E.A. Fox. Latent fingerprint segmentation using ridge template correlation. In *Imaging for Crime Detection and Prevention 2011 (ICDP 2011), 4th International Conference on*, pages 1–6, Nov 2011.

[125] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pages 958–, Washington, DC, USA, 2003. IEEE Computer Society.

[126] Kuldeep Singh, Anubhav Gupta, and Rajiv Kapoor. Fingerprint image super-resolution via ridge orientation-based clustered coupled sparse dictionaries. *Journal of Electronic Imaging*, 24:24 – 24 – 10, 2015.

[127] W. Strunk Jr. and E. B. White. *The Elements of Style*. Macmillan, New York, NY, 3rd edition, 1979.

[128] S. A. Sudiro, M. Paindavoine, and T. M. Kusuma. Simple fingerprint minutiae extraction algorithm using crossing number on valley structure. In *IEEE Workshop on Automatic Identification Advanced Technologies*, pages 41–44, June 2007.

[129] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[130] E. Tabassi, C. Wilson, and C. Watson. Fingerprint image quality. In *Technical Report 7151*, August 2006.

[131] Xuejun Tan, B. Bhanu, and Yingqiang Lin. Fingerprint classification based on learned features. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(3):287–300, Aug 2005.

[132] J. Tang, C. Deng, and G.B. Huang. Extreme learning machine for multilayer perceptron. 2015.

[133] Jiexiong Tang, Chenwei Deng, and Guang-Bin. Huang. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

[134] Yao Tang, Fei Gao, Jufu Feng, and Yuhang Liu. Fingernet: An unified deep network for fingerprint minutiae extraction. *CoRR*, abs/1709.02228, 2017.

[135] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014.

[136] Matej Ulicny and Rozenn Dahyot. On using CNN with DCT based image data. 2017.

[137] J. van der Geer, J. A. J. Hanraads, and R. A. Lupton. The art of writing a scientific article. *J. Sci. Commun.*, 163:51–59, 2000.

[138] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[139] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

[140] G. Wang and J. Qi. Penalized likelihood PET image reconstruction using patch-based edge-preserving regularization. *IEEE Transactions on Medical Imaging*, 31(12):2194–2204, Dec 2012.

[141] Yun-Heng Wang, Jiaqing Qiao, Jun-Bao Li, Ping Fu, Shu-Chuan Chu, and John F. Roddick. Sparse representation-based MRI super-resolution reconstruction. *Measurement*, 47:946 – 953, 2014.

[142] C. Watson, G. Fiumara, E. Tabassi, S. L. Cheng, P. Flanagan, and W. Salamon. Fingerprint vendor technology evaluation 2013: Summary of results and analysis report. NISTIR 8034, 2014, 2015.

[143] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016.

[144] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. *CoRR*, abs/1601.03651, 2016.

[145] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, Nov 2010.

[146] Z. Yang and M. Jacob. Nonlocal regularization of inverse problems: A unified variational framework. *IEEE Transactions on Image Processing*, 22(8):3192–3203, Aug 2013.

[147] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3466–3473, June 2012.

[148] Ilker Yildirim. Bayesian inference: Gibbs sampling. *Technical Note, University of Rochester*, 2012.

[149] S. Yoon, K. Cao, E. Liu, and A. K. Jain. Lfiq: Latent fingerprint image quality. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8, Sept 2013.

[150] J. Yu, B. Bhanu, and N. Thakoor. Face recognition in video with closed-loop super-resolution. In *CVPR 2011 WORKSHOPS*, pages 39–45, June 2011.

[151] Z. Yuan, J. Wu, S. Kamata, A. Ahrary, and P. Yan. Fingerprint image enhancement by super resolution with early stopping. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 4, pages 527–531, Nov 2009.

[152] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, June 2015.

[153] Jiangyang Zhang, Rongjie Lai, and C.-C.J. Kuo. Latent fingerprint segmentation with adaptive total variation model. In *Biometrics (ICB), 2012 5th IAPR International Conference on*, pages 189–195, March 2012.

[154] Lei Zhang and Xiaolin Wu. An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Transactions on Image Processing*, 15:2226–2238, 2006.

[155] Y. Zhu, X. Yin, X. Jia, and J. Hu. Latent fingerprint segmentation based on convolutional neural networks. In *IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Dec 2017.

[156] Weiwei Zong, Guang-Bin Huang, and Yiqiang Chen. Weighted extreme learning machine for imbalance learning. *Neurocomputing*, 101:229–242, 2013.

[157] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486, Nov 2011.

[158] X. Zou, X. Xu, C. Qing, and X. Xing. High speed deep networks based on discrete cosine transformation. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5921–5925, Oct 2014.