Chapter 4
Latent Fingerprint Image Segmentation Using Deep Neural Network

Jude Ezeobiejesi and Bir Bhanu

Abstract We present a deep artificial neural network (DANN) model that learns latent fingerprint image patches using a stack of restricted Boltzmann machines (RBMs), and uses it to perform segmentation of latent fingerprint images. Artificial neural networks (ANN) are biologically inspired architectures that produce hierarchies of maps through learned weights or filters. Latent fingerprints are fingerprint impressions unintentionally left on surfaces at a crime scene. To make identifications or exclusions of suspects, latent fingerprint examiners analyze and compare latent fingerprints to known fingerprints of individuals. Due to the poor quality and often complex image background and overlapping patterns characteristic of latent fingerprint images, separating the fingerprint region of interest from complex image background and overlapping patterns is very challenging. Our proposed DANN model based on RBMs learns fingerprint image patches in two phases. The first phase (unsupervised pre-training) involves learning an identity mapping of the input image patches. In the second phase, fine-tuning and gradient updates are performed to minimize the cost function on the training dataset. The resulting trained model is used to classify the image patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. The proposed model is evaluated by comparing the results from the state-of-the-art latent fingerprint segmentation models. The results of our evaluation show the superior performance of the proposed method.

J. Ezeobiejesi (✉) · B. Bhanu
Center for Research in Intelligent Systems, University of California at Riverside, Riverside, CA 92521, USA
e-mail: jezeobie@cs.ucr.edu

B. Bhanu
e-mail: bhanu@cris.ucr.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_4

4.1 Introduction

Deep learning is a technique for learning features using hierarchical layers of neural networks. There are usually two phases in deep learning. The first phase commonly referred to as pre-training involves unsupervised, layer-wise training. The second phase (fine-tuning) involves supervised training that exploits the results of the first phase. In deep learning, hierarchical layers of learned abstraction are used to accomplish high level tasks [3]. In recent years, deep learning techniques have been applied to a wide variety of problems in different domains [3]. Some of the notable areas that have benefited from deep learning include pattern recognition [21], computer vision [16], natural language processing, and medical image segmentation [17]. In many of these domains, deep learning algorithms outperformed previous state-of-the-art algorithms.

Latent fingerprints are fingerprint impressions unintentionally left on surfaces at a crime scene. Latent examiners analyze and compare latent fingerprints to known fingerprints of individuals to make identifications or exclusions of suspects [9]. Reliable latent fingerprint segmentation is an important step in the automation of latent fingerprint processing. Better latent fingerprint matching results can be achieved by having automatic latent fingerprint segmentation with a high degree of accuracy. In recent years, 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. Several high-profile cases in the United States and abroad have shown that forensic examiners can sometimes make mistakes when analyzing or comparing fingerprints [14] manually. Latent fingerprints have significantly poor quality ridge structure and large nonlinear distortions compared to rolled and plain fingerprints. As shown in Fig. 4.1, latent fingerprint images contain background structured noise such as stains, lines, arcs, and sometimes text. The poor quality and often complex image background and overlapping patterns characteristic of latent fingerprint images make it very challenging to separate the fingerprint regions of interest from complex image background and overlapping patterns [29]. To process latent fingerprints, latent experts manually mark the regions of interest (*ROIs*) in latent fingerprints and use the *ROIs* to search large databases of reference full fingerprints and identify a small number of potential matches for 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. As a step in this direction, this chapter proposes an efficient technique for separating latent fingerprints from the complex image background using deep learning. We learn a set of features using a hierarchy of RBMs. These features are then passed to a supervised learning algorithm to learn a classifier for patch classification. We use the result of the classification for latent fingerprint image segmentation. To the best of our knowledge, no previous work has used this strategy to segment latent fingerprints.

Fig. 4.1 Sample latent fingerprints from NIST SD27 showing three different quality levels **a** good, **b** bad, and **c** ugly

The rest of the chapter is organized as follows: Sect. 4.2.1 reviews recent works in latent fingerprint segmentation while Sect. 4.2.1.1 describes the contributions of this chapter. Section 4.3 highlights our technical approach and presents an overview of RBM as well discussion on learning with RBMs. The experimental results and performance evaluation of our proposed approach are presented in Sect. 4.4. Section 4.4.5 highlights the impacts of diffusing the training dataset with fractal dimension and lacunarity features on the performance of the network while Sect. 4.5 contains the conclusions and future work.

4.2 Related Work and Contributions

4.2.1 Related Work

Recent studies carried out on latent fingerprint segmentation can be grouped into three categories:

- Techniques based on classification of image patches
- Techniques based on clustering
- Techniques that rely on ridge frequency and orientation properties

The study presented in [9] falls into the first category. The authors performed image segmentation by extracting 8×8 nonoverlapping patches from a latent fingerprint image and classifying them into fingerprint and non-fingerprint patches using fractal dimension features computed for each image patch. They assembled the fingerprint patches to build the fingerprint portion (segmented region of interest) of the original image.

In the second category of approaches, Arshad et al. [4] used K-means clustering to divide the latent fingerprint image into nonoverlapping blocks and computed the standard deviation of each block. They considered a block as foreground if its standard

deviation is greater than a predefined threshold otherwise, it was a background block. They used morphological operations to segment the latent fingerprint.

The approaches that fall into the third category rely on the analysis of the ridge frequency and orientation properties of the ridge valley patterns to determine the area within a latent fingerprint image that contains the fingerprint [4, 7, 13, 27, 29]. Choi et al. [7] 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. [13] 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. [27] 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 six 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 evaluation criteria for the segmentation results. Zhang et al. [29] 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.

Our approach uses a deep architecture that performs learning and classification in a two-phase approach. The first phase (unsupervised pre-training) involves learning an identity mapping of the input image patches. In the second phase (fine-tuning), the model performs gradient updates to minimize the cost function on the dataset. The trained model is used to classify the image patches and the results of the classification are used for latent fingerprint image segmentation.

4.2.1.1 Contributions

This chapter makes the following contributions:

- Modification of how the standard RBM learning algorithm is carried out to incorporate a weighting scheme that enables the RBM in the first layer to model the input data with near zero reconstruction error. This enabled the higher level weights to model the higher level data efficiently.
- A cost function based on weighted harmonic mean of missed detection rate and false detection rate is introduced to make the network learn the minority class better, and improve per class accuracy. By heavily penalizing the misclassication of minority (fingerprint) class, the learned model is tuned to achieve close to zero missed detection rate for the minority class.

- The proposed generative feature learning model and associated classifier yield state-of-the-art performance on latent fingerprint image segmentation that is consistent across many latent fingerprint image databases.

4.3 Technical Approach

Our approach involves partitioning a latent fingerprint image into 8×8 nonoverlapping blocks, and learning a set of stochastic features that model a distribution over image patches using a generative multilayer feature extractor. We use the features to train a single-layer perceptron classifier that classifies the patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. The block diagram of our proposed approach is shown in Fig. 4.2, and the architecture of the feature learning, extraction, and classification model is shown in Fig. 4.3.

4.3.1 Restricted Boltzmann Machine

A restricted Boltzmann machine is a stochastic neural network that consists of visible layer, hidden layer, and a bias unit [11]. A sample RBM with binary visible and hidden units is shown in Fig. 4.4. The energy function E_f of RBM is linear in its free parameters and is defined as [11]:

$$E_f(\hat{x}, h) = -\sum_i b_i \hat{x}_i - \sum_j c_j h_j - \sum_i \sum_j \hat{x}_i w_{i,j} h_j, \tag{4.1}$$

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 probability distributions over visible or hidden vectors are defined in terms of the energy function [11]:

$$P(\hat{x}, h) = \frac{1}{\omega} e^{-E_f(\hat{x}, h)}, \tag{4.2}$$

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 [11] and is defined as:

$$P(\hat{x}) = \frac{1}{\omega} \sum_h e^{-E_f(\hat{x}, h)} \tag{4.3}$$

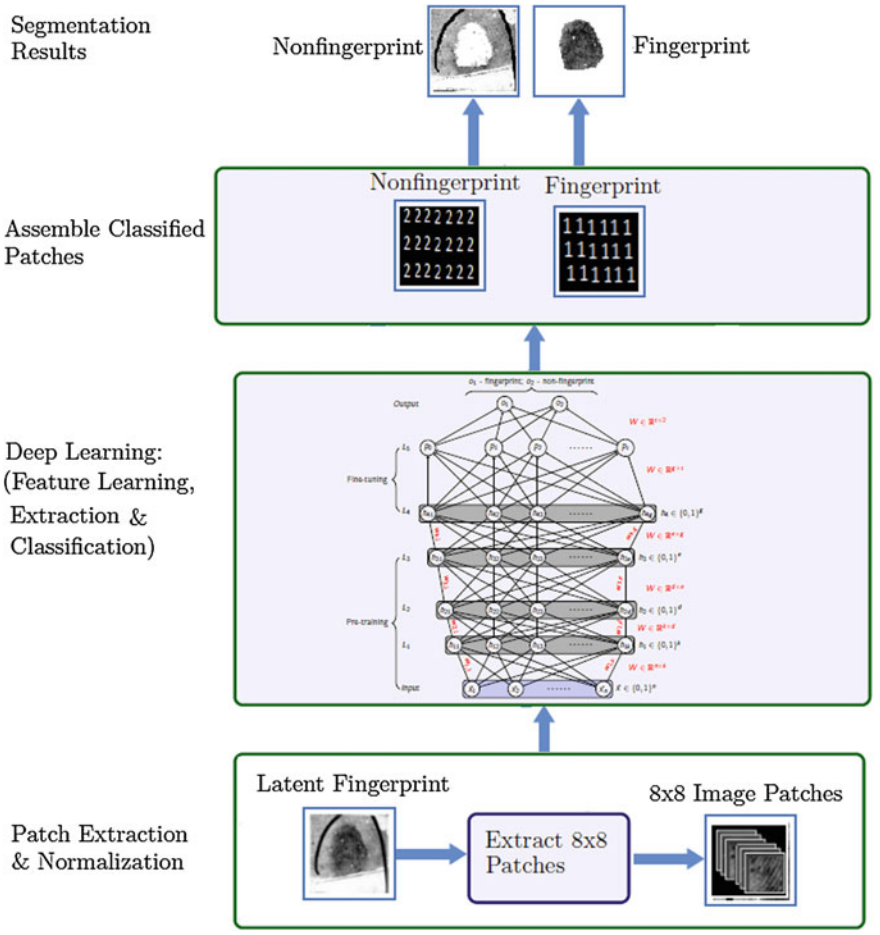

Fig. 4.2 Proposed approach

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 [6]. The conditional probability of a configuration of the visible units is given by

$$P(\hat{x}|h) = \prod_{i=1}^{n} P(\hat{x}_i|h), \tag{4.4}$$

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

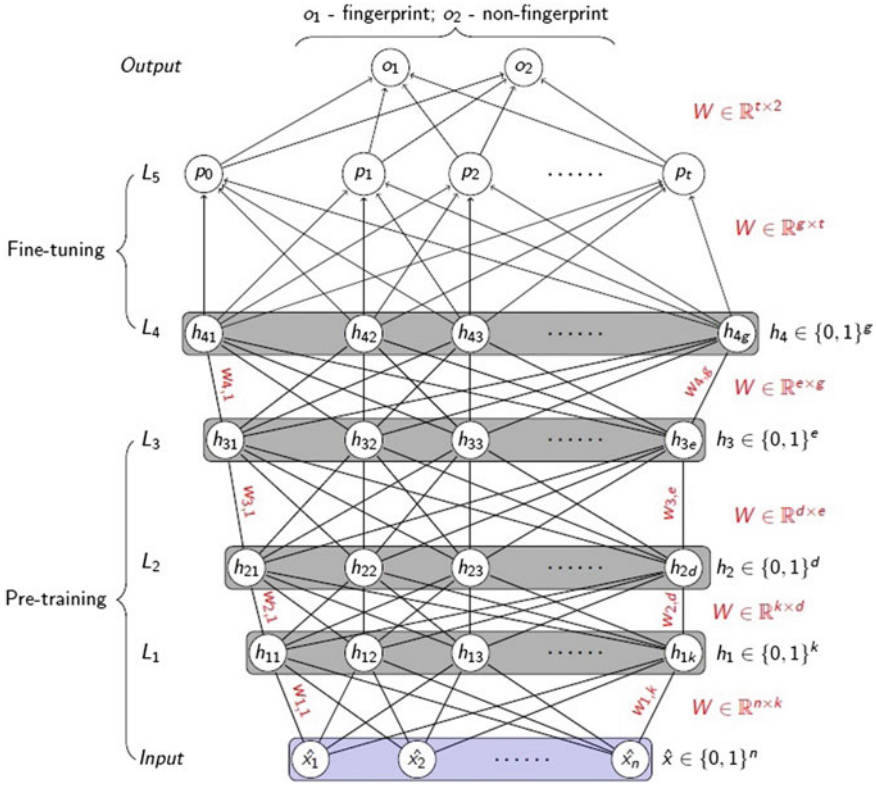

Fig. 4.3 Feature learning, extraction, and classification using a multilayer neural network. The pre-training phase uses the input layer (visible units), and three hidden layers of RBM (L_1, L_2, L_3). The fine-tuning phase uses an RBM layer (L_4) and a single-layer perceptron (L_5). The output layer has two output neurons (fingerprint and non-fingerprint). All the units are binary. $h_{i,j}$ is the jth node in L_i, $w_{i,j}$ is the weight connecting the ith node in layer L_i to the jth node in layer L_{i-1}. We set $n = 81$ (64 from 8×8 and 17 from diffusion), $k = 800$, $d = 1000$, $e = 1200$, $g = 1200$, $t = 1200$, where n, k, d, e, g, t are the number of nodes in the input layer, L_1, L_2, L_3, L_4, L_5, respectively

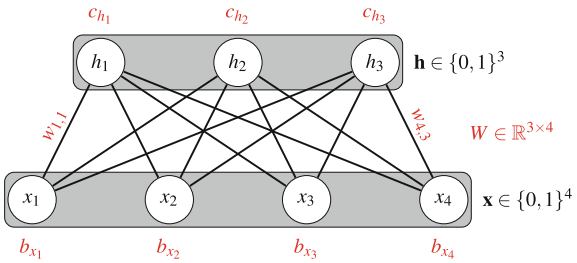

Fig. 4.4 Graphical depiction of RBM with binary visible and hidden units. x_i, $i = 1, \ldots, 4$, are the visible units while h_k, $k = 1, \ldots, 3$, are the hidden units. b_{x_i}, $i = 1, \ldots, 4$, are the biases for the visible units and c_{h_k}, $k = 1, \ldots, 3$, are the biases for the hidden units

$$P(h|\hat{x}) = \prod_{j=1}^{m} P(h_j|\hat{x}), \tag{4.5}$$

where m is the number of hidden units. The individual activation probabilities are given by

$$P(h_j = 1|\hat{x}) = \sigma\left(b_j + \sum_{i=1}^{n} w_{i,j}\hat{x}_i\right) \tag{4.6}$$

and

$$P(\hat{x}_i = 1|h) = \sigma\left(c_i + \sum_{j=1}^{m} w_{i,j}h_j\right), \tag{4.7}$$

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

4.3.2 Learning with RBM

Learning with RBM involves several steps of sampling hidden variables given visible variables, sampling visible variables given hidden variables, and minimizing reconstruction error by adjusting the weights between the hidden unit and visible layers. The goal of learning with RBM is to identify the relationship between the hidden and visible variables using a process akin to identity mapping. We performed the sampling step using Gibbs sampling technique enhanced with contrastive divergence.

4.3.2.1 Gibbs Sampling

A sampling algorithm based on Monte Carlo Markov Chain (MCMC) technique used in estimating desired expectations in learning models. It allows for the computation of statistics of a posterior distribution of given simulated samples from that distribution [28]. A Gibbs sampling of the joint of R random variables $R = (R_1, R_2, \ldots, R_n)$ involves a sequence of R sampling sub-steps of the form $R_i \sim p(R_i|R_{-i})$ where R_i contains the *n-1* other random variables in R excluding R_i. For RBMs, $R = Q_1 \cup Q_1$ where $Q_1 = \{\hat{x}_i\}$ and $Q_2 = \{h_i\}$. Given that the sets Q_1 and Q_2 are conditionally independent, the visible units can be sampled simultaneously given fixed values of the hidden units using block Gibbs sampling. Similarly, the hidden units can be sampled simultaneously given the visible units. The following is a step in the Markov chain:

$h^{(k+1)} \sim \sigma(W^T v^{(k)} + c)$

$\hat{x}^{(k+1)} \sim \sigma(W h^{(k+1)} + b)$,

where $h^{(k)}$ refers to the set of all hidden units at the kth step of the Markov chain and σ denotes logistic sigmoid defined as

$$o(x) = \frac{1}{1 + e^{-W_v z(x) - b}} \tag{4.8}$$

$$\text{with } z(x) = \frac{1}{1 + e^{-W_h x - c}}, \tag{4.9}$$

where W_h and c are the weight matrix and bias for the hidden layers excluding the first layer, and $z(x)$ is the activation of the hidden layer in the network. W_v is a weight matrix connecting the visible layer to the first hidden layer, and b is a bias for the visible layer.

4.3.2.2 Contrastive Divergence (CD-k)

This algorithm is used inside gradient descent procedure to speed up Gibbs sampling. It helps in optimizing the weight W during RBM training. CD-k speeds up Gibbs sampling by taking sample after only k-steps of Gibbs sampling, without waiting for the convergence of the Markov chain. In our experiments we set $k = 1$.

4.3.2.3 Stochastic Gradient Descent

With large datasets, computing the cost and gradient for the entire training set is usually very slow and may be intractable [24]. This problem is solved by Stochastic Gradient Descent (SGD) by following the negative gradient of the objective function after seeing a few training examples. SGD is used in neural networks to mitigate the high cost of running backpropagation over the entire training set [24].

Given an objective function $J(\phi)$, the standard gradient descent algorithm updates the parameters ϕ as follows:

$$\phi = \phi - \alpha \nabla_\phi E[J(\phi], \tag{4.10}$$

where the expectation $E[J(\phi]$ is obtained through an expensive process of evaluating the cost and gradient over the entire training set. With SGD, the gradient of the parameters are computed using a few training examples with no expectation to worry about. The parameters are update as,

$$\phi = \phi - \alpha \nabla_\phi J(\phi; x^{(i)}, y^{(i)}) \tag{4.11}$$

where the pair $(x^{(i)}, y^{(i)})$ are from the training set. Each parameter update is computed using a few training examples. This reduces the variance in the parameter update with the potential of leading to more stable convergence. Prior to each training epoch, we

randomly shuffled the training data to avoid biasing the gradient. Presenting the training data to the network in a nonrandom order could bias the gradient and lead to poor convergence.

One of the issues with learning with stochastic gradient descent is the tendency of the gradients to decrease as they are backpropagated through multiple layer of nonlinearity. We worked around this problem by using different learning rates for each layer in the proposed network.

4.3.2.4 Cost Function

Our goal is to classify all fingerprint patches (minority class) correctly to meet our segmentation objective of extracting the region of interest (fingerprint part) from the latent fingerprint image. We introduced a cost function based on the weighted harmonic mean of missed detection rate and false detection rate. We adopted a weight assignment scheme that was skewed in favor of the minority class to make the neural network learn the minority class better. Given a set of weights $w_1, w_2, \ldots, w_n$ associated with a dataset $x_1, x_2, \ldots, x_n$, the weighted harmonic mean $\mathbb{H}$ is defined as

$$\mathbb{H} = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}} = \left(\frac{\sum_{i=1}^{n} w_i x_i^{-1}}{\sum_{i=1}^{n} w_i} \right)^{-1}. \tag{4.12}$$

By penalizing the misclassification of minority class more, the model learned to detect minority class with a high degree of accuracy. The cost function is defined as:

$$\mathbb{C} = \frac{2}{\frac{1}{\tau MDR} + \frac{1}{\tau FDR}}, \tag{4.13}$$

where τMDR and τFDR are the weighted missed detection rate and weighted false detection rate, respectively. They are computed as: $\tau MDR = \tau_1 * MDR$ and $\tau FDR = \tau_2 * FDR$, where $\tau_1 = \frac{P_s + N_s}{P_s}$ and $\tau_2 = \frac{P_s + N_s}{N_s}$ are the weights assigned to positive class samples P_s and negative class samples N_s, respectively.

Table 4.1 shows a comparison of the error cost during the fine-tuning phase of our model with cross entropy cost function, and the proposed cost function.

4.3.3 Choice of Hyperparameters

We selected the value of the hyperparameters used in the proposed network based on the performance of the network on the validation set. The parameters and their values are shown in Table 4.2.

Table 4.1 Comparison of model performance using regular cost function (cross entropy) and proposed cost function. The mean, maximum, and minimum error costs are better (smaller is better) with the proposed cost function. With the proposed cost function, the model is tuned to achieve a low missed detection rate

Cost function	Min. error cost	Max. error cost	Mean error cost
Cross entropy	3.53E-03	1.041E+00	6.29E-02
Proposed	6.00E-04	1.10E-02	2.03E-03

Table 4.2 Parameters and values

Parameter	L_0	L_1	L_2	L_3	L_4	L_5	L_6
Number of neurons	81	800	1000	1200	1200	1200	2
Batch size	–	100	100	100	100	100	–
Epochs	–	50	50	50	50	–	–
Learning rate	–	1e-3	5e-4	5e-4	5e-4	–	–
Momentum	–	0.70	0.70	0.70	0.70	–	
Number of iterations	–	–	–	–	–	50	–

4.3.3.1 Unsupervised Pre-training

We adopt unsupervised layer-wise pre-training because of its power in capturing the dominant and statistically reliable features present in the dataset. The output of each layer is a representation of the input data embodying those features. According to [8], greedy layer-wise unsupervised pre-training overcomes the challenges of deep learning by introducing a useful prior to the supervised fine-tuning training procedure. After pre-training a layer, its input sample is reconstructed and the mean square reconstruction error is computed. The reconstruction step entails guessing the probability distribution of the original input sample in a process referred to as generative learning. Unsupervised pre-training promotes input transformations that capture the main variations in the dataset distribution [8]. Since there is a possibility that only a small subset of these variations may be relevant for predicting the class label of a sample, using a small number of nodes in the hidden layers will make it less likely for the transformations necessary for predicting the class label to be included in the set of transformations learned by unsupervised pre-training. This idea is reflected in our choice of the number of nodes in the pre-training layers. We ran several experiments to determine the optimal nodes in each of the three pre-training layers. As shown in Table 4.2, the number of nodes in the pre-training layers L_1, L_2, and L_3 are 800, 1000, and 1200, respectively.

4.3.3.2 Supervised Fine-Tuning

Supervised fine-tuning is the process of backpropagating the gradient of a classifier's cost through the feature extraction layers. Supervised fine-tuning boosts the performance of neural networks with unsupervised pre-training [19]. In our model, supervised fine-tuning is done with a layer of RBM and a single-layer perceptron depicted as L_4 and L_5, respectively in Fig. 4.3. During the fine-tuning phase, we initialized L_4, with the pre-trained weights of the top-most pre-training layer L_3.

4.4 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.

4.4.1 Latent Fingerprint Databases

We tested our model on the following databases:
• **NIST SD27:** This database was acquired from the National Institute of Standards and Technology. It contains images of **258** latent crime scene fingerprints and their matching rolled tenprints. The images in the database are classified as good, bad, or ugly based on the quality of the image. The latent prints and rolled prints are at **500** ppi.
• **WVU Database:** This database is jointly owned by West Virginia University and the FBI. It has **449** latent fingerprint images and matching **449** rolled fingerprints. All images in this database are at **1000** ppi.
• **IIITD Database:** The IIITD was obtained from the Image Analysis and Biometrics lab at the Indraprastha Institute of Information Technology, Delhi, India [25]. There are **150** latent fingerprints and **1,046** exemplar fingerprints. Some of the fingerprint images are at **500** ppi while others are at **1000** ppi.

4.4.2 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 and is defined as.

$$MDR = \frac{FN}{TP + FN} \tag{4.14}$$

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 fingerprint patches.It is defined as

$$FDR = \frac{FP}{TN + FP} \tag{4.15}$$

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.

$$SA = \frac{TP + TN}{TP + FN + TN + FP} \tag{4.16}$$

4.4.3 Stability of the Architecture

To investigate the stability of the proposed architecture, we performed five runs of training the network using 50,000 training samples. All the model parameters (number of epochs, number of iterations etc.) shown in Table 4.2 remained unchanged across the runs. The mean square reconstruction error (msre), mean error cost, and standard deviation for the five runs are shown in Table 4.3. Plots of the reconstruction errors against number of training epochs as well as that of error cost against number or iterations during each run are shown in Fig. 4.5. These results show that our model is stable.

Table 4.3 Network Stability: The msre, error cost, MDR, FDR, and training accuracy for the five different runs are close. The mean and standard deviation indicate stability across the five runs

Run #	MSRE	Error cost	MDR	FDR	Training accuracy
1	0.0179	5.469e-04	2.010e-04	0.00	4.00e-05
2	0.0183	5.406e-04	3.020e-04	0.00	6.00e-05
3	0.0178	5.560e-04	1.010e-04	0.00	2.00e-05
4	0.0179	5.438e-04	2.020e-04	0.00	5.00e-05
5	0.0178	6.045e-04	1.010e-04	0.00	2.00e-05
Mean	0.0179	5.584e-04	1.814e-04	0.00	3.800e-05
Standard deviation	0.0002	2.643e-05	8.409e-05	0.00	1.789e-05

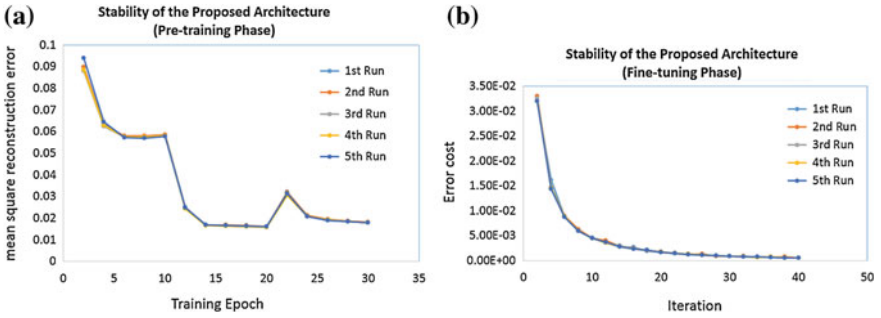

Fig. 4.5 Network Stability: **a** shows that the mean square reconstruction error (msre) followed the same trajectory during the five different runs, converging close to 0.02% msre. Similarly, **b** shows that the error cost during the fine-tuning phase followed the same trajectory for the five runs, converging to about 5.5E-04 error cost. The results are indicative of the stability of the network

4.4.4 Segmentation Using the Trained Network

To segment a latent fingerprint image using the trained network we proceed as follows:

- Split the image into 8×8 nonoverlapping patches and augment each patch data with its fractal dimension and lacunarity features to create a segmentation dataset.
- Normalize the segmentation dataset to have 0 mean and unit standard deviation.
- Load the trained network and compute activation value for each neuron: $a = \sum Wx$
- Feed the activation value to the activation function to normalize it.
- Apply the following thresholding function to obtain the classification results:

$$\theta(x) = \begin{cases} 1 : z > T \\ 0 : z \leq T \end{cases} \tag{4.17}$$

where z is the decision value, x is an example from the segmentation dataset, T is a threshold that gave the best segmentation accuracy on a validation set and was obtained using fivefold cross validation described in Algorithm 1.

4.4.4.1 Searching for Threshold *T*

We implemented a *hook* into the logic of output neurons to access the real-valued output of the activation function. To obtain the percentage of the activation function output for a given neuron, we divided its activation function value by the sum of all activation function values. For each label $y \in 1, 2$, we ordered the validation examples according to their decision values (percentages) and for each pair of adjacent decision values, we checked the segmentation accuracy using their average as T. The algorithm used was inspired by [10], and is shown as Algorithm 1.

Algorithm 1 Searching for threshold T

procedure THRESHOLD(X, Y) ▷ X is the patch dataset, Y is a set of corresponding labels
  $num\_class \leftarrow Unique(Y)$ ▷ Get unique labels from Y
  for $cs = 1, \ldots, num\_class$ **do** ▷ Iterate over the number of classes
    (a) $folds \leftarrow Split(X, 5)$ ▷ Split the validation set into five folds
    for $f = 1, \ldots, folds$ **do** ▷ Iterate over the folds
      (i) Run Compute(.) on four folds of validation set ▷ Run four folds through the trained network
      (ii)$T_c^f \leftarrow Best()$ ▷ Set T_c^f to the decision value that achieved the best *MDR*
    (b) Run Compute(.) on X ▷ Run the validation set through the trained network
    $T_c \leftarrow \frac{1}{5} \sum_{k=1}^{folds} T_c^k$ ▷ Set the threshold to the average of the five thresholds from cross validation
  return T ▷ Return the threshold

4.4.5 Dataset Diffusion and Impact on Model Performance

Given a dataset $X = \{x_1, x_2, \ldots, x_k\}$, we define the diffusion of X as $\hat{X} = \{x_1, x_2, \ldots, x_m\}$, where $m > k$ and each $x_i, k < i < m$ is an element from R^n. In other words, $\hat{X}$ is obtained by *expanding* X with new elements from R^n. 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 [12, 22]. We used features based on fractal dimension and lacunarity to diffuse X. These features help to characterize complex texture in latent fingerprint images [9].

4.4.5.1 Fractal Dimension

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 [23] and was first used in texture analysis by Keller et al. [15]. Fractal dimension offers a quantitative way to describe and characterize the complexity of image texture composition [18].

We compute the fractal dimension of an image patch P using a variant of differential box counting (DBC) algorithm [2, 26]. 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 nonoverlapping $d \times d$ 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 $h = \frac{\mathscr{T} d}{N}$, where $\mathscr{T}$ is the total gray levels in P, and d is an integer. Let $\mathscr{T}_{min}$ and $\mathscr{T}_{max}$ be the minimum and maximum gray levels in grid *(i, j)*, respectively. The number of boxes covering block *(i, j)* is given by:

$$n_d(i,j) = floor[\frac{\mathscr{T}_{max} - \mathscr{T}_{min}}{r}] + 1, \tag{4.18}$$

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:

$$N_d = \sum_{i,j} n_d(i,j) \tag{4.19}$$

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.

$$FD_P = \sum_{i=1,j=1}^{MN} FD_{ij} \tag{4.20}$$

♦ **Fractal Dimension Features**: We implemented a variant of the *DBC* algorithm [2, 26], to compute the following statistical features from the fractal dimension image P'.

• ***Average Fractal Dimension***:

$$FD_{avg} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} FD_{ij} \tag{4.21}$$

• ***Standard Deviation of Fractal Dimension:*** 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

$$FD_{std} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} (FD_{ij} - FD_{avg})^2, \tag{4.22}$$

• ***Fractal Dimension Spatial Frequency:*** 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 [20]. 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

$$R_f = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=1}^{N-1} [G(x,y) - G(x,y-1)]^2} \tag{4.23}$$

$$C_f = \sqrt{\frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=1}^{N-1}[G(x,y) - G(x-1,y)]^2} \tag{4.24}$$

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

$$FD_{sf} = \sqrt{R_f^2 + C_f^2} \tag{4.25}$$

From signal processing perspective, Eqs. (4.23) and (4.24) favor high frequencies and yield values indicative of patches with fingerprint.

4.4.5.2 Lacunarity

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 [5]. Lacunarity (FD_{lac}) is defined in terms of the ratio of variance over mean value [2].

$$FD_{lac} = \frac{\frac{1}{MN}(\sum_{i=1}^{M-1}\sum_{j=1}^{N-1}P(i,j)^2)}{\{\frac{1}{MN}\sum_{i=1}^{M-1}\sum_{j=1}^{N-1}P(i,j)\}^2} - 1, \tag{4.26}$$

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

4.4.5.3 Diffusing the Dataset

We followed standard engineering practice to select the architecture of our model. To improve the performance of the model, we tried various data augmentation techniques such as label preserving transformation and increasing/decreasing the number 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.

Due to discriminative capabilities of fractal dimension and lacunarity features, we used them to diffuse the patch dataset. From experiments, we observed that by diffusing the dataset with these features before normalizing the data 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 Fig. 4.6. As can be seen from Table 4.4, when the training dataset was augmented with FD features, there was a huge drop in both error cost during fine-tuning and the classification error during

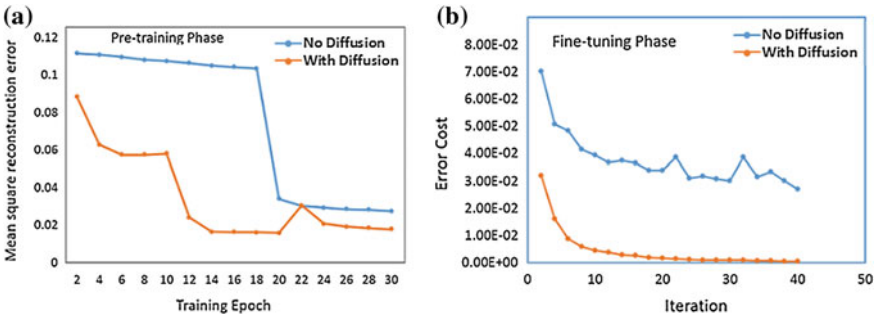

Fig. 4.6 Impact of Data Diffusion on Model Performance. **a** shows that during the pre-training phase, the network achieves lower mean square reconstruction error (msre) when the dataset is diffused with fractal dimension features. Also, as can be seen from **b**, diffusing the dataset leads to faster convergence and lower error cost during the fine-tuning phase

Table 4.4 Data diffusion and network performance

	MSRE	Error cost	Classification error (Training) (%)
Without diffusion	0.0179	7.97e-01	18.51
With diffusion	0.0178	6.0456e-04	0.006

training. It is interesting to note that the reconstruction error almost remained the same in both cases.

4.4.6 Classification and Segmentation Results

4.4.6.1 Training, Validation, and Testing

We studied the performance of our model when trained on one latent fingerprint database and tested on another using 3 sets of 20,000 patches, 40% drawn from good, 30% from bad, and 30% from ugly images from NIST, WVU, and IIITD databases. In each of the three experiments, 10,000 patches from a set were used for training, 4,000 for validation, and 6,000 for testing. The results are shown in Table 4.5.

The final training, validation and testing of the model was done with 233,200 patches from the NIST SD27 database with 40% from good, 30% from bad, and 30% from ugly NIST image categories. 132,000 examples were used for training, 48,000 for validation, and 53,200 for testing. Table 4.6 shows the confusion matrix for NIST SD 27 and Table 4.7 shows the TP, TN, FP, and FN, MDR, FDR and classification accuracy on the training, validation, and testing datasets. There was no

Table 4.5 Model performance when trained and tested on different latent fingerprint databases. The numbers in bracket delimited with colon are the training, validation, and testing datasets, respectively. The three datasets are independent. The training and validation datasets shown in column 1 of the last row were obtained exclusively from NIST SD27 database. The testing sets are independent of the training set and were obtained from the target testing database in column 5. MDR_V and FDR_V are the validation MDR and FDR, respectively. Similarly, MDR_T and FDR_T are the testing MDR and FDR, respectively. As shown in the last row, there was a marked improvement in the model performance when more training data was used. When we tried more than 132,000 patches for training, there was no appreciable performance gain despite more training time required to achieve convergence

Train on	Validate on	MDR_V (%)	FDR_V (%)	Test on	MDR_T (%)	FDR_T (%)
NIST SD27 (10,000 : 4,000 : 6,000)	NIST SD27	2.95	1.92	NIST SD27	3.04	1.98
				WVU	3.75	2.25
				IIITD	3.63	2.19
WVU (10,000 : 4,000 : 6,000)	WVU	3.12	2.54	NIST SD27	3.61	3.01
				WVU	3.22	2.87
				IIITD	3.90	3.05
IIITD (10,000 : 4,000 : 6,000)	IIITD	3.32	2.66	NIST SD27	3.49	3.19
				WVU	3.86	3.16
				IIITD	3.28	2.80
NIST SD27 (132,000 : 48,000 : 53,200)	NIST SD27	1.25	0	NIST SD27	1.25	0
				WVU	1.64	0.60
				IIITD	1.35	0.54

Table 4.6 NIST SD27—Confusion matrix for training, validation, and testing

		Predicted patch class (Training)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	23,667	9
	Non-Fingerprint	0	108,324
		Predicted patch class (Validation)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	12,946	154
	Non-Fingerprint	0	34,900
		Predicted patch class (Testing)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	15,291	193
	Non-Fingerprint	0	37,716

Table 4.7 NIST SD27—Training, Validation and Testing Accuracy: Training: **132,000** 8×8 patches; Validation: **48,000** 8×8 patches; Testing: **53,200** 8×8 patches. $MDR = \frac{FN}{TP+FN}$; $FDR = \frac{FP}{TN+FP}$

	TP	TN	FP	FN	MDR (%)	FDR (%)	Classification accuracy (%)
Training	23,667	108,324	0	9	0.04	0	99.96
Validation	12,946	34,900	0	154	1.17	0	98.83
Testing	15,291	37,716	0	193	1.25	0	98.75

noticeable performance gain when the model was trained with more than 132,000 patches.

4.4.6.2 Segmentation Results

Figures 4.7, 4.8, and 4.9 show the segmentation results of our proposed method on sample good, bad, and ugly quality images from the NIST SD27 database. The figures show the original latent fingerprint images and the segmented fingerprints and non-fingerprints constructed using patches classified as fingerprints and non-fingerprints.

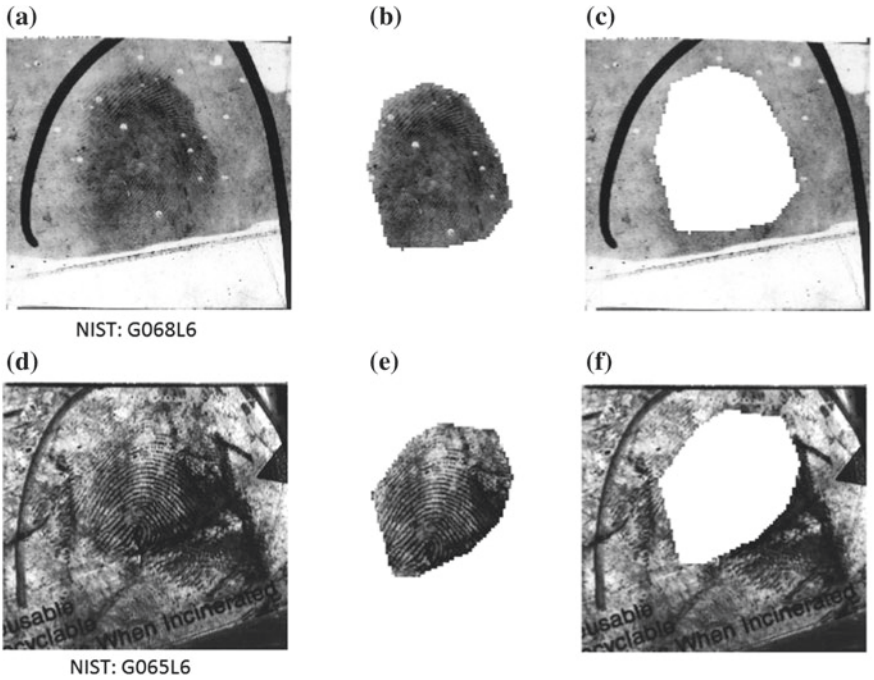

Fig. 4.7 NIST Good Category Latent fingerprint image and segmentation result without post classification processing. **a** and **d** Original images **b** and **e** Fingerprints **c** and **f** Non-fingerprints

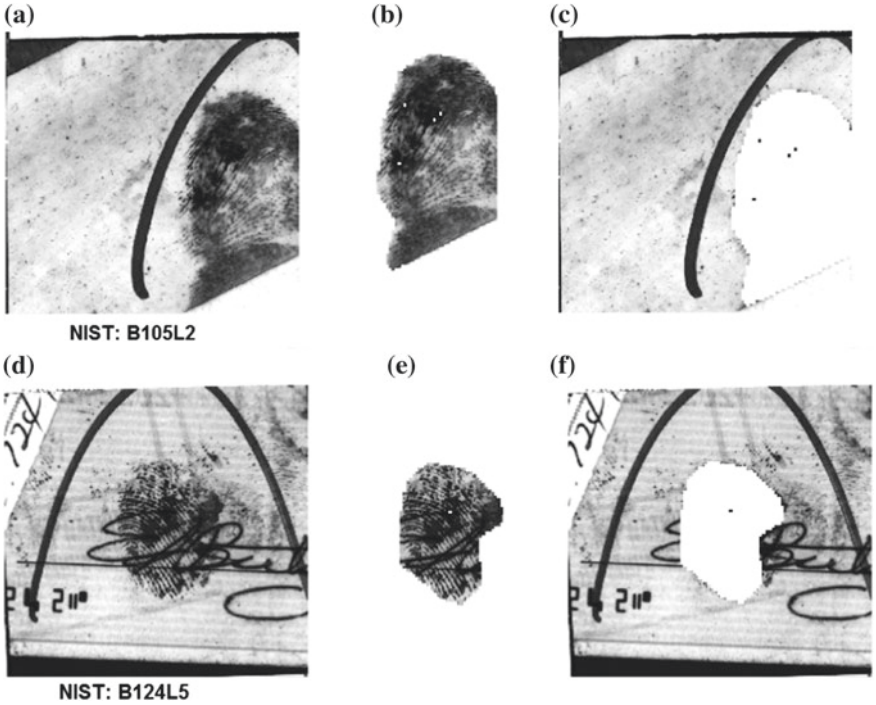

Fig. 4.8 NIST Bad Category [1] Latent Fingerprint Image and segmentation result without post classification processing. **g** and **j** Original images; **h** and **k** Fingerprints **i** and **l** Non-fingerprints

The segmentation results for WVU and IIITD are not shown due to restrictions in the database release agreement (Fig. 4.10).

4.4.7 Comparison with Current Algorithms

Table 4.8 shows the superior performance of our segmentation approach on the good, bad, and ugly quality latent fingerprints from NIST SD27 compared to the results from existing algorithms on the same database. It also shows the performance comparison of our model on WVU and IIITD with other algorithms that reported results on those latent fingerprint databases.

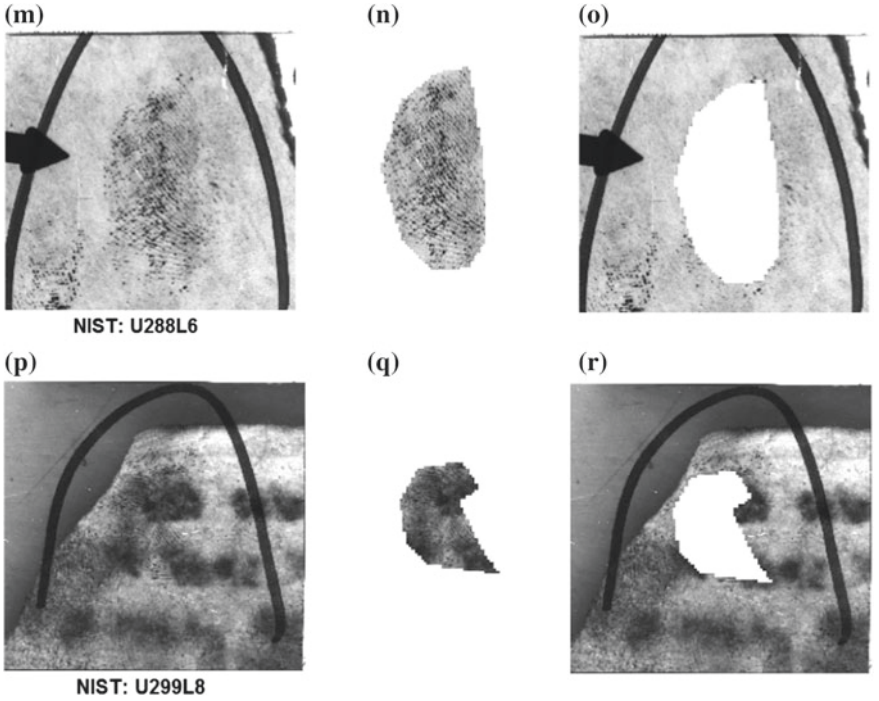

Fig. 4.9 NIST Ugly Category Latent Fingerprint Image and segmentation result without post classification processing. **m** and **p** Original images **n** and **q** Fingerprints **o** and **r** Non-fingerprints

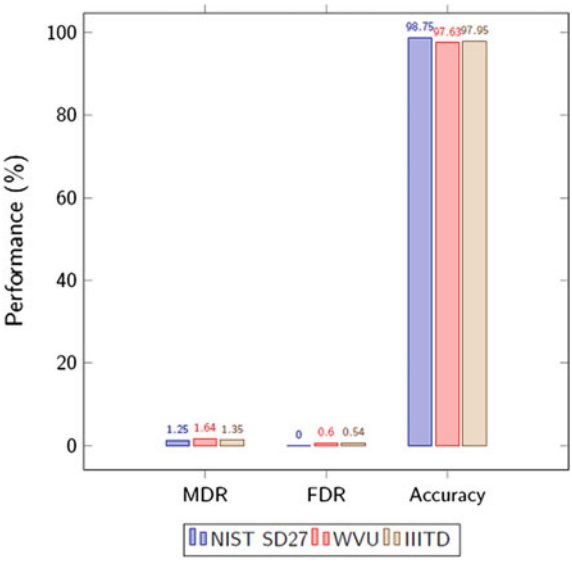

Fig. 4.10 Segmentation reliability in different databases for good quality images. This shows the results of training our model on NIST SD27 and testing on NIST SD27, WVU, and IIITD latent databases. The choice of latent fingerprint database used during training has small impact on the performance of our network. This assertion is also supported by the results in Table 4.5

Table 4.8 Comparison with other algorithms on various datasets

Author	Approach	Database	MDR %	FDR %	Average
Choi et al. [7]	Ridge orientation and frequency computation	NIST SD27	14.78	47.99	31.38
		WVU LDB	40.88	5.63	23.26
Zhang et al. [29]	Adaptive total variation model	NIST SD27	14.10	26.13	20.12
Arshad et al. [4]	K-means clustering	NIST SD27	4.77	26.06	15.42
Jude and Bhanu [9]	Fractal dimension & Weighted ELM	NIST SD27 (Good, Bad, Ugly)	9.22	18.7	13.96
		WVU LDB (Good, Bad, Ugly)	15.54	9.65	12.60
		IIITD LDB (Good)	6.38	10.07	8.23
This chapter	Deep learning	NIST SD27 (Good, Bad, Ugly)	1.25	0.04	0.65
		WVU LDB (Good, Bad, Ugly)	1.64	0.60	1.12
		IIITD (Good)	1.35	0.54	0.95

4.5 Conclusions and Future Work

We proposed a deep architecture based on restricted Boltzmann machine for latent fingerprint segmentation using image patches and demonstrated its performance on the segmentation of latent fingerprint images. The model learns a set of stochastic features that model a distribution over image patches. Using the features extracted from the image patches, the model classifies the patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. We demonstrated the performance of our model in the segmentation of good, bad, and ugly latent fingerprints from the NIST SD27, as well as WVU and IIITD latent fingerprint databases. We showed that the overall performance of our deep model is superior to that obtained with the state-of-the-art latent fingerprint image segmentation algorithms. Our future work involves developing algorithms for feature extraction and matching for the segmented latent fingerprints.

Acknowledgements This research was supported in part by the Presley Center for Crime and Justice Studies, University of California, Riverside, California, USA.

References

1. NIST Special Database 27. *Fingerprint Minutiae from Latent and Matching Ten-print Images*, http://www.nist.gov/srd/nistsd27.htm
2. O. Al-Kadi, D. Watson, Texture analysis of aggressive and nonaggressive lung tumor CE CT images. IEEE Trans. Biomed. Eng. **55**(7), 1822–1830 (2008)
3. I. Arel, D.C. Rose, T.P. Karnowski, Deep machine learning - a new frontier in artificial intelligence research [research frontier]. IEEE Comput. Intell. Mag. **5**(4), 13–18 (2010)
4. I. Arshad, G. Raja, A. Khan, Latent fingerprints segmentation: feasibility of using clustering-based automated approach. Arabian J. Sci. Eng. **39**(11), 7933–7944 (2014)
5. M. Barros Filho, F. 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)
6. M.A. Carreira-Perpinan, G. Hinton, On contrastive divergence learning, in *AISTATS*, vol. 10 (Citeseer, 2005), pp. 33–40
7. H. Choi, A.I.B.M. Boaventura, A. Jain, Automatic segmentation of latent fingerprints, in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2012), pp. 303–310
8. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)
9. J. Ezeobiejesi, 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* (2016)
10. R.-E. Fan, C.-J. Lin, *A Study on Threshold Selection for Multi-label Classification*. Department of Computer Science (National Taiwan University, 2007), pp. 1–23
11. G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines, Version 1 (2010)
12. C. Huang, C. Moraga, A diffusion-neural-network for learning from small samples. Int. J. Approx. Reason. **35**(2), 137–161 (2004)
13. S. Karimi-Ashtiani, C.-C. Kuo, A robust technique for latent fingerprint image segmentation and enhancement, in *15th IEEE International Conference on Image Processing, 2008, ICIP 2008* (2008), pp. 1492–1495
14. D. Kaye, T. Busey, M. Gische, G. LaPorte, C. Aitken, S. Ballou, L.B. ..., K. Wertheim, Latent print examination and human factors: improving the practice through a systems approach, in *NIST Interagency/Internal Report (NISTIR) - 7842* (2012)
15. J. Keller, S. Chen, R. Crownover, Texture description and segmentation through fractal geometry. Comput. Vis. Graph. Image Process. **45**(2), 150–166 (1989)
16. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
17. M. Lai, Deep learning for medical image segmentation (2015), arXiv:1505.02000
18. A.D.K.T. Lam, Q. Li, Fractal analysis and multifractal spectra for the images, in *2010 International Symposium on Computer Communication Control and Automation (3CA)*, vol. 2 (2010), pp. 530–533
19. P. Lamblin, Y. 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)
20. S. Li, J.T. Kwok, Y. Wang, Combination of images with diverse focuses using the spatial frequency. Inf. Fus. **2**(3), 169–176 (2001)
21. Y. Liu, E. Racah, P.J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, W.D. Collins, Application of deep convolutional neural networks for detecting extreme weather in climate datasets (2016), CoRR abs/1605.01156
22. Z. Makó, Approximation with diffusion-neural-network, in *6th International Symposium of Hungarian Researchers on Computational Intelligence* (2005), pp. 18–19

23. B. Mandelbrot, *The Fractal Geometry of Nature*. Einaudi paperbacks (Henry Holt and Company, New York, 1983)
24. A. Ng, J. Ngiam, C.Y. Foo, http://ufldl.stanford.edu/tutorial/supervised/optimizationstochasticgradientdescent/, *UFLDL Tutorial*
25. A. Sankaran, M. Vatsa, R. Singh, Hierarchical fusion for matching simultaneous latent fingerprint, in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2012), pp. 377–382
26. N. Sarkar, B.B. Chaudhuri, An efficient differential box-counting approach to compute fractal dimension of image. IEEE Trans. Syst. Man Cybern. **24**(1), 115–120 (1994)
27. N. Short, M. Hsiao, A. Abbott, E. Fox, Latent fingerprint segmentation using ridge template correlation, in *4th International Conference on Imaging for Crime Detection and Prevention 2011 (ICDP 2011)* (2011), pp. 1–6
28. I. Yildirim, Bayesian inference: Gibbs sampling. Technical Note, University of Rochester (2012)
29. J. Zhang, R. Lai, C.-C. Kuo, Latent fingerprint segmentation with adaptive total variation model, in *2012 5th IAPR International Conference on Biometrics (ICB)* (2012), pp. 189–195