Fingerprint Image Enhancement Using Unsupervised Hierarchical Feature Learning

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science and Engineering

by

Mihir Sahasrabudhe
200802023
mihir.s@research.iiit.ac.in

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2015

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Fingerprint Image Enhancement Using Unsupervised Hierarchical Feature Learning" by Mihir Sahasrabudhe, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. Anoop Namboodiri

To my family.

Acknowledgements

I couldn't have completed this work without the support, help, guidance, and company of several people in my life. First and foremost, I would like to send my sincerest thanks to my adviser, Dr. Anoop Namboodiri. He is the most humble and down-to-earth professor I have ever met. He has been a guiding light in all matters, not only academic, and has always been available and helpful whenever I found myself stuck on problems. He has helped me immensely in determining a direction of research, which has been a source of inspiration for further studies. I would also like to thank Rama Reddy K N V for helping me kick-start my research. His enthusiasm and patience in teaching me was pivotal. I would also like to thank Siddhartha Chandra for helping me grasp deep learning. I have always found learning from a teacher more efficient and less tedious than learning from a book, and that was realised here.

I would like to thank Dr. Abhijit Mitra and his family for helping me adjust to life at IIIT-Hyderabad. We had probably the most varied and diverse discussions together.

I send thanks to the professors in CVIT - Dr. Jawahar, Dr. PJN, and Dr. Jayanthi, for driving the research here. I thank Satya sir for helping out in all administrative matters. I would also like to thank my lab-mates - Aditya, Ankit, Chandrasekhar, Harshit, Jay, Mayank, Parikshit, Prabhu, Rajvi, Revanth, Rohan, Siddhartha, Srinath, Saurabh, Ujjwal, Akhil, Aniket, Ayush, Divyansh, Yash - with whom I exchanged ideas, discussed topics, and shared laughs and stories.

I thank my band-mates at *Karmic Blend* - Sankalp, Siddhartha, Pulkit, Soumen. We shared a common passion for, and taste of music. I thoroughly enjoyed all the practice sessions, live shows, and the dinners we had together.

I thank my friends, fellow dual-degree students and partners in crime - Dabba, Darinda, Dubey, Fugga, Gandhi, Kharaab, Masala, Tondua, Majay, Dosa, Vyas, Speaker Dhillon, Cracky, Singhal, Moidu, Reddy, Fakka, Chhutka, Macchi, Thaki. The post-graduate years at IIIT-Hyderabad are one of my most treasured memories, thanks to them.

Last but certainly not the least, I would like to thank my family. Any amount of thanks will not be enough for the constant support and encouragement they have given me throughout my research; always ensuring I wouldn't digress; keeping me on the right path. They have shown incredible patience with my Masters research, and guided me in important decisions I had to take with respect to future commitments.

Abstract

The use of fingerprints is an important method for identification of individuals in today's world. They are also one of the most reliable biometric traits, besides the iris. Fingerprint recognition refers to various tasks that are associated with fingerprint identification, verification, feature extraction, indexing, enhancement and classification. There are a lot of systems, in a variety of domains, that employ fingerprint recognition. That being a given, precision in fingerprint recognition is essential.

Identification using fingerprints is done using a feature extraction step, followed by matching of these features. The features that are extracted to be matched depend on the algorithm being used for identification. In large databases of fingerprints, like government records, fingerprints might be indexed before they are matched. This significantly reduces the time required to identify an individual from records, as comparing his/her prints with every entry in the database will take a enormous amount of time. In either case, feature extraction plays an important role. However, feature extraction is affected directly by the quality of the input image. A noisy or unclear fingerprint image might affect the extraction of features strongly. To counter noise in input images, a step of enhancement is introduced before feature extraction and matching are performed. The goal of extraction is to improve quality of ridges and valleys in the fingerprint by making them clearly distinguishable, but in the process, also preserve information. The enhancement algorithm should not only not omit or remove existing information from the fingerprint, but also not introduce any spurious features that were not present in the original image.

The considerable research into fingerprint recognition, and in fingerprint enhancement, has contributed to the large number of existing algorithms for image enhancement. These include pixel-wise enhancement, contextual filtering, and short-time Fourier analysis, to name a few. Contextual filtering uses specific filters to convolve the input image with, the filters themselves being governed by certain parameters dependent on the input image. These parameters are determined by the values of certain features at every pixel in the input image. This requires extraction of pre-defined features from the fingerprint. For instance, the filter used at a point is affected by the ridge orientation at that point. Hence, to decide the filter to be used, the ridge orientation at that point needs to be extracted. A similar case can be observed in other types of algorithms too.

In this thesis, we propose that unsupervised feature learning be applied to the fingerprint enhancement problem. We use two different scenarios and models to show that unsupervised feature learning indeed helps improve an existing algorithm, and also when applied directly to greyscale images, can complete with robust contextual filtering and Fourier analysis algorithms. Our motivation lies in the fact

that there is vast amount of available data on fingerprints; and with the recent advances in deep learning, and unsupervised feature learning in particular, we can use this available data to learn structures in fingerprint images.

For the first model, we show that continuous restricted Boltzmann machines can be used to learn local fingerprint orientation field patterns, after which their learning can be used to correct noisy, local ridge orientations. This extra step is introduced between orientation field estimation and contextual filtering. We show that this step improves the performance of matching done on the enhanced images. In the second model, we use a 2-layered convolutional deep belief network to learn features directly from greyscale fingerprint images. We show that having a deep neural network significantly improves the quantitative and qualitative performance of the enhancement. The deep network helps in predicting noisy regions, that were otherwise not reconstructed by the first layer only. Further, a trained convolutional deep belief network can estimate or extract other features from fingerprint images too. For instance, the orientation field used to determine the parameters of a filter used in contextual filtering is a feature that can be estimated by the neural network. This is possible by performing a weighted average of the values of these features for the first layer filters, weighted by the reconstruction performed by the network.

In conclusion, we have explored a new direction to attack the fingerprint enhancement problem. We conjecture that it is possible to extend this work to other problems involving fingerprint recognition too. For instance, synthetic fingerprint generation might be accomplished using the convolutional deep belief network trained on fingerprint features. Our experiments show potential, which opens up several potential experiments for the future which can give promising results. Future work also includes developing or training a single network that is capable of performing major fingerprint recognition tasks: enhancement, matching, and classification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's world, the task of identifying a person carries much importance. Be it for access to e-mail or a secure facility, identification of a person/suspect in an investigation, creating a central database for citizens of a country, or many more such jobs, we require systems that can efficiently and robustly identify and verify individuals. We can point out such systems quite frequently in our day-to-day life. Fingerprint scanners on laptops and in classrooms serve, respectively, the purpose of identifying an individual and granting access, and verifying an individual to mark their presence. Further, security holds a major share in the employment of such systems. While the fingerprint scanner in the classroom isn't used to keep anything secure, a scanner at a secure facility certainly does. The large stake held by security here is a driving factor for the requirement of high accuracy of these systems. Depending on what level of security is required and what kind of systems are feasible, a method of identification is chosen.

1.1 Biometric Recognition

We saw that systems that can identify a person are an essential requirement in today's technology. There are several ways to achieve this identification - passwords, word of a trusted authority, biometric traits are a few of them. A biometric trait is an anatomical and/or behavioural trait of a person, which can be used to identify him/her. For instance, in every day life, we identify people primarily based on their face; sometimes we identify them with their voice too. When we see a person walking towards us from far away, we figure out who that person is by looking at how he/she walks. Face, voice, and gait are thus some examples of biometric traits. There are some which are not used by people in every day life, but they can identify a person. Examples of such traits are fingerprints, iris of the eye, the retina and the vein-pattern on the hands. Employment of biometric traits for automatic recognition or verification of individuals is called biometric recognition, or simply, biometrics.

Of the many available biometric identifiers, the fingerprint is the one which offers the highest score on the following requirements, which are deemed essential for a biometric to be used to identify a person.

1. **Universality**: The biometric trait should be possessed by everyone.

2. **Distinctiveness**: It should be sufficiently distinct for every person.

3. **Permanence**: The trait should not fade/change/disappear over time.

4. **Collectability**: It should be easily collectible.

A biometric system is a system that recognises an individual based on one or many of the available biometric traits. A biometric system may be classified into the following two categories:

- **Identification system**: An identification system is a system which takes as input(s) a person's biometric trait(s), and scans its database to find a match for these traits, hence identifying the person based on the traits.

- **Verification system**: A verification system is a system which takes as input(s), biometric trait(s) of a person, compares them with the records in its database, and verifies whether the said person really is the one he/she is claiming to be.

Biometric systems have been built which use fingerprint as the biometric trait to verify/identify individuals. However, besides the above properties for a biometric trait, we also have to consider the following factors when building a biometric system:

1. **Performance**: Performance encompasses factors such as computation time, running time, accuracy, robustness and resource requirements.

2. **Acceptability**: This defines the extent to which users would be willing to accept the biometric system.

3. **Circumvention**: The ease with which such a system can be circumvented.

Given in table 1.1 is a comparison of the various biometric traits, and their suitability for each of the factors mentioned above (Taken from *Handbook of Fingerprint Recognition* [47], by Maltoni, Maio, Jain and Prabhakar).

This table suggests fingerprints have high potential of being parameters for biometric identification.

1.2 Fingerprints as Reliable Biometric Identifiers

A fingerprint is an inherent pattern on the skin of a person's finger. It has been observed that a fingerprint remains consistent throughout the lifetime of a person. Even if the finger suffers from cuts or bruises, the fingerprint reappears after it has healed. This gives fingerprints a very high score on permanence. Further, fingerprints have been demonstrated to be unique to every individual, and hence can serve as an identification parameter. Fingerprints are being used as an identification parameter for over

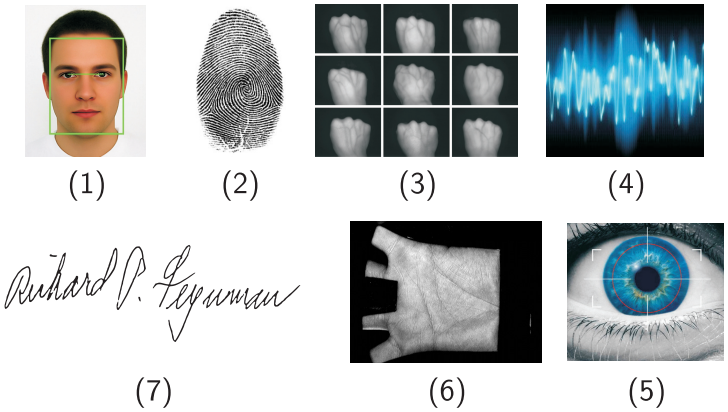

Figure 1.1 Examples of biometric traits used for identification and verification of individuals. Clockwise from top-left: (1) face; (2) fingerprint; (3) hand veins; (4) voice; (5) iris; (6) palmprint; and (7) signature.

a century now, with the first effort being made by Alphonse Bertillon. The uniqueness of fingerprints was accepted in 1893 by the Home Ministry Office of the UK. Few anomalies have been found in the 100+ years of study of fingerprints. By this observation, it is safe to assign a high score to fingerprints on distinctiveness. Until recently, verification and identification tasks using fingerprints were primarily being performed by experts who were well versed with the processes. That is to say, there weren't automated systems that would perform these tasks. The advent of computers and the use of computers and computer-systems for identification and verification is a fairly recent phenomenon, compared to the time for which fingerprints have been around as identification parameters. This has certainly made the process more convenient, less cumbersome and much faster, and with the tremendous strides achieved by researchers in fingerprint recognition, we can safely assign a score of high to fingerprints on the 'performance' metric. However, fingerprints as biometrics don't perform so well on other primary metrics:

1. **Universality.** Not every person on earth has a fingerprint, unlike the face, which is possessed by almost any one who is alive. Some people lose their prints, while some others aren't born without them. Examples of this include severe burns on hands, amputations, and being born without limbs.

2. **Collectability.** Fingerprints, although easily available, aren't as easily collectible. They cannot be collected using the common cameras. Collection of fingerprints requires special sensors, the availability, suitability and usability of which drags down the rating on this metric.

3. **Acceptability.** As they aren't a traditional means of identification of an individual (traditionally, faces or signatures would suffice), fingerprints aren't very accepted by the general public. Further, recording a fingerprint can be a bit of a hassle, given that one might need to dirty their hands or deal with faulty or dirty sensors.

Trait	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention	Score
Fingerprint	M	H	H	M	H	M	M	27
Iris	H	H	H	M	H	L	H	27
Face	H	L	M	H	L	H	L	23
Hand Geometry	M	M	M	H	M	M	M	23
Hand Vein	M	M	M	M	M	M	H	21
Signature	L	L	L	H	L	H	L	17
Voice	M	L	L	M	L	H	L	17

Table 1.1 A comparison of various biometric traits on primary requirements. The total score is calculated by assigning H a score of 5, M a score of 3 and L a score of 1. Instead of rating the ease of circumvention, the last column rates the difficulty in getting around the system. High scores indicate higher difficulty in breaking the system (Taken from *Handbook of Fingerprint Recognition* [47]).

4. **Circumvention.** Getting around a system that uses fingerprints to authenticate isn't as hard as it may seem. We leave our fingerprints almost everywhere during our day-to-day lives. It is extremely easy for an impostor to collect the fingerprint left by us, although a little difficult to fool a sensor into thinking that the fingerprint shown to it is authentic.

1.3 Fingerprint Features

As is the case with any data, the features that can be extracted from fingerprints are crucial. Feature extraction forms an integral part of all fingerprint recognition tasks: enhancement, alignment, matching, and classification. There are four primary features that are helpful in achieving these tasks. These will be discussed to a considerable extent in this section.

However, before we move to that, we introduce definitions for ridges and valleys in a fingerprint image. The inherent pattern present on the epidermis is made up of gaps and protrusions. These protrusions, as we shall see in Section 1.4, result in the black lines in the resultant fingerprint image. These protrusions are called ridges. Also, the gaps between ridges result in the white area in the resultant image. These gaps are called valleys. Figure 1.2 illustrates this nomenclature in a better manner.

Now that we have established definitions for ridges and valleys, we can move to defining features in fingerprint images. We will look at four categories: ridge orientation, ridge frequency, singularities, and behaviours of ridges. As expected, all of these features are of how ridges and valleys appear at a point in the fingerprint image.

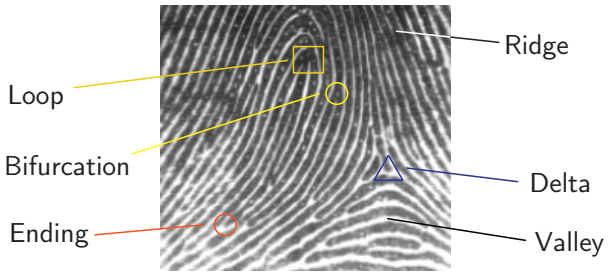

Figure 1.2 A fingerprint image showing a ridge, a valley, a ridge ending (red), a ridge bifurcation (yellow), a loop (green), and a delta (blue). Ridges cause the black pixels in the captured fingerprint image, whereas the valleys don't leave marks in the image.

1.3.1 Orientation

Orientation, as the name might suggest, refers to the direction of ridge-flow at a point. More precisely, the orientation at a point (x, y) in a fingerprint image is the angle made by the ridges with the horizontal in an arbitrarily small neighbourhood centered at (x, y) (Figure 1.3). Further, an *orientation field*, $\mathbf{D}$, is the image formed by setting the value at every pixel to be equal to the orientation at that pixel. Figure 1.4 illustrates the orientation field.

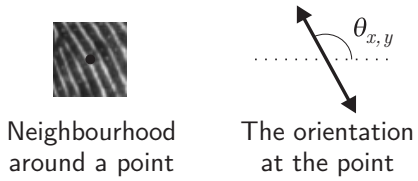

Figure 1.3 The orientation at a point. The arrow heads indicate that the orientation is independent of towards which of the two ends we consider the ridge-flow to be and hence, orientation values of θ and $\pi + \theta$ are equivalent.

As the ridges are not directional (for example, we don't say whether a horizontal ridge is flowing to the left or to the right), an orientation value of θ is equivalent to a value of the form $k\pi + \theta$. This implies that the orientation lies in the range $[0, \pi[$ (The notation $[x, y[$ says that this range folds over itself, i.e., $y + t \equiv x + t$). Further, a reliability measure is also generally included with the orientation. Reliability measure at a point tells us the quality of the fingerprint region at that point. High quality regions have a high reliability measure, whereas low quality regions have a low measure. One way to determine the reliability is to calculate the *coherence* of the orientation field in that region. Kass and Witkin [34] determine the coherence as the norm of sum of the orientation vectors divided by the sum of norms of the vectors:

$$r_{xy} = \text{coherence}_{xy} = \frac{\left|\sum_W \mathbf{d}\right|}{\sum_W |\mathbf{d}|} \tag{1.1}$$

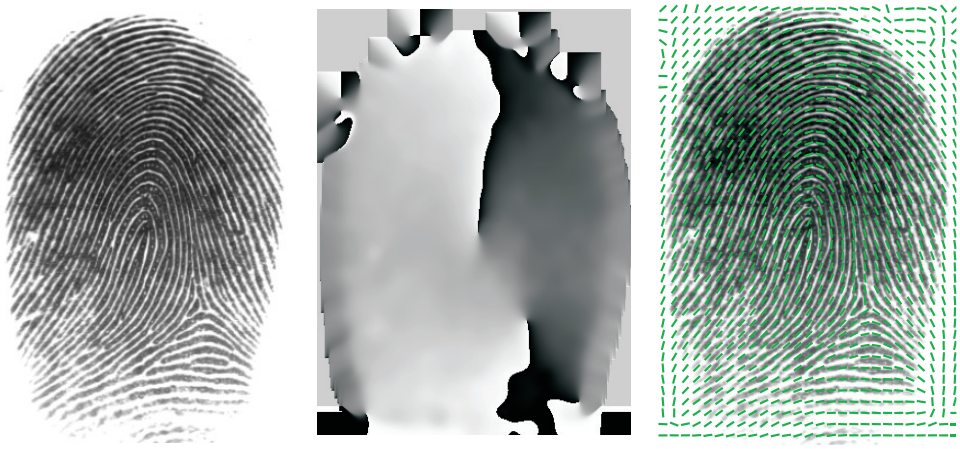

Figure 1.4 *left*: The original fingerprint image; *centre*: the orientation field illustrated as an image; and *right*: the orientation field depicted as a plot of orientated lines, the directions of which are determined by the values of the orientations at pixels.

Here, W is a neighbourhood around the point (x, y). $\mathbf{d}$ (defined in the next section) is an orientation vector in the neighbourhood W. The thinking behind this is simple: as fingerprint ridges are continuous in a small neighbourhood, orientations at all points in this neighbourhood should be approximately in the same direction. In such a case, the norm of the sum of the orientation vectors would be close to the sum of norms of individual orientation vectors. Orientation vectors in the window W around a point (x, y) in a noisy region are expected to be not parallel to each other (parallel vectors would generate the most coherence). Hence, the reliability for a noisy region drops, as the value of the numerator in Equation 1.1 drops.

Calculating the orientation field. Over the years, several methods have been suggested to estimate the orientation field of a fingerprint image. Gradient-based estimation is most common. It functions on the fact that the direction of maximum gradient of pixel values at a point is perpendicular to the orientation at that point. Hence, to estimate the orientation at a point, we first compute the x and y gradients at that point. The orientation is then simply $\pi/2 + \arctan(\nabla_y/\nabla_x)$, where ∇_y and ∇_x are, respectively, the gradients along y and x directions. However, this is susceptible to even minor inconsistencies in pixel values, which are to be expected as fingerprint ridges aren't perfectly regular. Also, this presents discontinuities around an orientation of $\pi/2$, as it discards the circularity of orientation angles. Kass and Witkin [34] proposed a work-around for this by introducing a small change in the way orientations were calculated, where an orientation vector would be represented by 2θ, instead of θ. This adjusts the circularity of angles:

$$\mathbf{d} = (r\cos(2\theta), r\sin(2\theta)) \tag{1.2}$$

r, here, is proportional to the length of the gradient vector, i.e., $\nabla_x^2 + \nabla_y^2$. By averaging orientations in an $n \times n$ window W, we get a more robust estimation:

$$\mathbf{d} = \left(\frac{1}{n^2} \sum_W r \cos(2\theta), \frac{1}{n^2} \sum_W r \sin(2\theta) \right) \tag{1.3}$$

Two further similar methods for estimating the orientation field are slit- and the projection-based methods. In these methods, the orientation values are first quantised, i.e., possible orientation values for every point in the image are fixed as multiples of a fraction of π. Hence,

$$\mathbf{D}(x, y) = \arg\min_k \left[\text{cost} \left(k \frac{\pi}{n_s} \right) ; \text{for } k = 0, 1, \ldots, n_s - 1 \right] \tag{1.4}$$

To find the orientation at a point, all of these possible orientation values are tested according to a "cost" function. Lower the cost of an orientation at a point, higher the preference for it at that point. One way is to compute standard deviations of pixel values along a slit and perpendicular to the slit, and choose the setting which gives the highest difference between standard deviations along these two slits [54]. Using projections, we can also find the direction of smallest variance of the pixel values. The direction along which the projection has the least variance corresponds to the direction of the ridge [64].

Instead of working in the spatial domain to estimate the orientation, we can also work in the frequency domain. It is interesting to note that a fingerprint patch resembles a 2-dimensional surface, which behaves like a sine wave in one direction. Kamei and Mizoguchi [32, 31] proposed a method for enhancement using directional filters in the frequency domain. The orientation at a point is determined by the highest filter response and local smoothing. Chikkerur *et al.* proposed the use of Short-time Fourier Transform [14] to estimate the orientation at a point. In this method, a sliding window operation is applied on the fingerprint image, and for each window, an orientation is estimated from the power spectrum of the Fourier transform of that region. Even further approaches to estimate the orientation field including smoothing orientation fields locally and modelling a global orientation field structure using MRFs.

1.3.2 Frequency

The ridge frequency at a point (x, y) in a fingerprint image is defined as the number of ridges per pixel in the direction perpendicular to the orientation at that point. The frequency image, like the orientation field, is an image where every pixel gets a value equal to the ridge orientation at that point. It is denoted by $\mathbf{F}$. Usually, ridge frequency varies from pixel to pixel in a fingerprint. Sometimes, this variation can be considerable. We can expect large variations between images of different fingers.

The local orientation is useful in determining the ridge frequency at a point. Hong *et al.* [47] devised a method which counted the average distance between ridges in a 32×16 oriented window. By oriented window, we mean a rectangular window with its longer side perpendicular to the ridge orientation. In this window, the x-signature is computed, which is the signal representing the sum of pixel values along

the columns of the rectangle. The ridge frequency is then the average distance between two peaks. However, peaks can be ambiguous at times. Hence, some processing needs to be done before the x-signature can be used. This includes interpolation and low-pass filtering. Another method to estimate the frequency at a point is to work in the frequency domain, just like measuring the orientation.

1.3.3 Singularities

Singularities, also called level 1 features, are features that can be observed at the broadest level of the fingerprint. These are formed by particular ridge structures and curvatures, and hence do not concern themselves with the structure of individual ridges. Singularities can be broadly classified into three types: loops, deltas, and whorls. However, sometimes whorls aren't included as a different singularity as they can be represented by two loops (Figure 1.5)

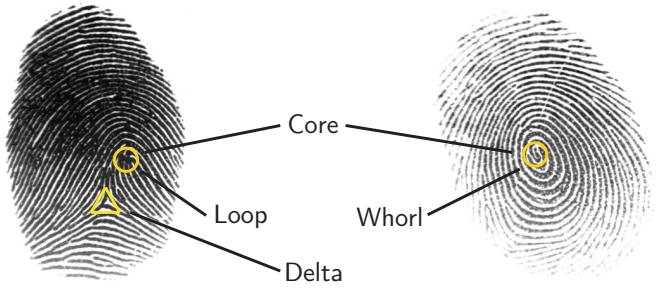

Figure 1.5 Examples of whorls, deltas and loops in fingerprint images. Loops and whorls have a common feature: the core.

1.3.4 Minutiae

Minutiae are points of interest on a fingerprint. They are characterised by behaviour of particular interest of ridges. Such behaviour can be classified into several types (Figure 1.6). The most commonly occurring minutiae in fingerprints are ridge endings, ridge bifurcations, independent ridges, lakes, spurs, islands, and crossovers. Of primary interest amongst these are ridge endings and ridge bifurcations. As the name suggests, they represent, respectively, an abrupt ending and a division into two of a ridge. The others can be broken down into a combination of these two. For example, a lake can be broken down as two closely spaced bifurcations, a spur can be looked at as a ridge bifurcation followed by a ridge ending, and an independent ridge is two closely spaced ridge endings.

Minutiae are level 2 features. Sir Francis Galton [20] was the first to recognise the presence of minutiae, and their permeability. They are sometimes called "Galton details", to honour him. A minutia is represented by three parameters: the x- and y-coordinates of the point where the minutia is present, and the orientation, θ, associated with it. This forms a triplet: $\langle x, y, \theta \rangle$. The orientation associated with a ridge ending is the orientation at the point of the ending. Determining the orientation for a bifurcation, however, can be ambiguous. Hence, a convention is followed which not only removes this ambiguity,

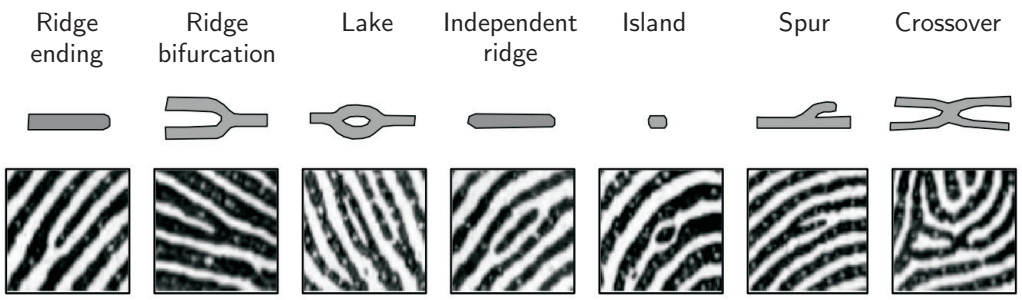

Figure 1.6 The most commonly found minutiae in fingerprints. The most commonly used, however, are only the first two, as the others can be decomposed as a combination of these two (Image taken from *Handbook of Fingerprint Recognition [47]).*

but also introduces a tolerance towards minutiae that are incorrectly classified. Figure 1.7 illustrates how θ is defined.

Following this convention, the angle associated with a bifurcation is equal to the orientation of the *negative ridge ending* of that bifurcation. Here, a negative ridge implies the ridge structure that is obtained by flipping over the pixel values of the image (hence transforming ridges into valleys, and vice versa). At a bifurcation, the negative ridge will show an ending, the orientation of which is easily defined. It is observed that sometimes ridge endings and ridge bifurcations are confused in multiple impressions of the same finger. This might happen due to different finger pressure at the time of taking the print, which might disrupt the connection between the two ridges in a ridge bifurcation, thus making it look like a ridge ending. This convention is tolerant towards such a scenario, and this convention doesn't introduce a considerable inconsistency in the associated orientation.

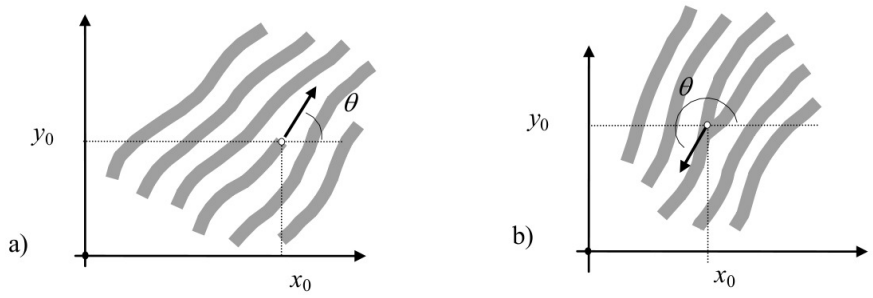

Figure 1.7 Minutiae angles associated with (a) ridge endings and; (b) bifurcations. The angle associated with bifurcation is the angle the negative ridge is oriented at (taken from *Handbook of Fingerprint Recognition [47]*).

1.4 Fingerprint Sensing

The process of gathering a digital copy of the imprint of a person's finger is referred to as fingerprint sensing. For a majority of the time period in which fingerprints have been used as methods as of

identification and verification of individuals, the task of fingerprint sensing was performed by gathering an inked print and scanning it to convert it into a digital copy. To gather an inked print, the hands of the person whose prints needed to be gathered were smeared in ink, and he/she was asked to place them on a sheet of paper. Next, this copy was digitised to be used by a computer by scanning it at a comfortable resolution, usually 500 dots per inch (dpi). However, as technology progressed and requirements evolved, a fast method to gather fingerprints was required. This gave rise to fingerprint sensors, which are electronic devices to register fingerprints. The process of recording of fingerprints is called enrolment. Hence, a person is said to be enrolled if he/she has provided his/her fingerprints to go on the record. There are several reasons for the need of enrolment. For instance, governments can link records of citizens with their fingerprints, secure facilities will need authorised persons' fingerprints to go on the record, so that they can be identified and verified within the facility, and personal computers, phones and other devices might have fingerprint sensors which also record the owner's prints to allow them access without a password. Sensors, hence, are spread across the scale of the task for which they are required. This emphasises the need for an efficient, easy-to-use, and robust device that can accomplish this task. Figure 1.8 is an example of the working of sensors.

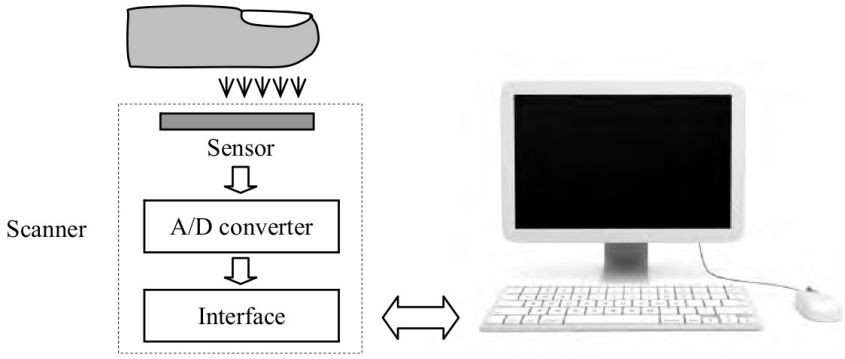

Figure 1.8 An example of a fingerprint sensor that outputs digital images of collected fingerprints. There is an analogue-to-digital converter applied on the sensed image so that it can be directly used by a computer. Some sensors don't require this converter as the image scanned by them is already digital. Image taken from *Handbook of Fingerprint Recognition* [47].

For each of the three scenarios mentioned here, taking gathering an inked fingerprint of the person in consideration, digitising it, and then verifying or identifying the individual is not feasible. Apart from being a great inconvenience to the involved parties, it is a time-consuming arrangement. If we use an electronic sensor instead, the prints can be gathered and digitised in a matter of seconds. This reduces the time taken by and effort required to identify someone several fold.

The fingerprint that is enrolled can be either a plain impression, or a rolled impression. A plain impression is obtained by pressing down a part of the finger on the sensor. This records only a portion of the fingerprint. A plain impression doesn't have complete information of a fingerprint. For instance, for it to match another plain impression of the same finger, the overlap between these two prints should be substantial. The two prints can't be matched at all if they have zero overlap. A rolled impression,

Figure 1.9 A plain impression (left) and a rolled impression (right) of the same finger. The part of the rolled impression highlighted corresponds to the plain one. Image taken from *Handbook of fingerprint recognition* [47].

on the other hand, is a "nail-to-nail" impression of a fingerprint. As the name suggests, it is obtained by "rolling" a finger on the sensor from one end to another, so that as much information is gathered as possible. Figure 1.9 gives an example of plain and rolled impressions.

Existing fingerprint scanners can be broadly classified into two classes: *multi-finger* scanners, and *single-finger* scanners. As the name suggests, multi-finger scanners have the ability to scan prints of multiple fingers at a time. These are usually used in systems which require mass enrolment of people - registering of citizens of a country by a government, for instance. The enrolment procedure is completed in three steps using a multi-finger scanner: all fingers (excluding the thumb) of one hand are scanned at once, then the same is done for the other hand, and finally, the two thumbs are scanned. However, the scanner doesn't generate a separate image for every finger that was scanned. Hence, a post-processing algorithm is included with the process which finds fingerprints, distinguishes them from each other, and assigns a finger to every fingerprint.

Figure 1.10 Some examples of fingerprint scanners. *left*: A single-finger scanner; *centre*: A multi-finger scanner; and *right*: A single-finger sweep scanner. Unlike the touch scanner, the sweep scanner asks the user to swipe his/her finger over it. This gives a sequence of overlapping slabs of the fingerprint image, which the scanner then stitches together.

Unlike multi-finger scanners, single-finger scanners can scan only one fingerprint at a time. This minimalist design results in a lower cost of the device as compared to multi-finger scanners. They can

hence be used at a non-commercial and personal level. They can be found on laptops to authenticate users, and outside classrooms to identify individuals for attendance.

Further, there are several methods to build a fingerprint sensor. The most common sensors can be classified as optical, thermal, capacitive, and ultrasound. The working of a general optical sensors involves flashing light on the surface of the sensor once a finger is placed on it. As only the ridges will touch the surface, they absorb most of the light that falls on them. The valleys, on the other hand, reflect all the light off the surface, as the light is incident so as to cause total internal reflection. The reflected light is gathered and an image is produced, which has different responses for valleys and ridges. Capacitive sensors work by forming small capacitors between the skin of the finger, and capacitive plates placed beneath the sensor-surface. As only the ridges touch the surface, the capacitance developed between a plate and a valley is different from the one developed between a plate and a ridge. From the resultant capacitive patterns, an inference is drawn of the fingerprint. Thermal sensors also rely on the fact that ridges touch the surface and valleys don't. A thermal sensor is heated up so that the surface of the sensor is at a high temperature. Next, when a finger is placed on it, the portion of the fingerprint that has ridges produce a different temperature than the region with valleys, and hence generates the resultant fingerprint pattern. The ultrasound sensor is has a transmitter and a receiver of sound waves. Operating on a concept similar to that of SONAR, these sensors sense ridges and valleys based on the sound reflected off them. Figure 1.11 shows some scanned fingerprints.

Figure 1.11 Fingerprints scanned using different types of scanners. From left to right: optical, capacitive, thermal and ultrasound. The thermal fingerprint was sensed using a sweep method, instead of a touch method. The sweep method requires the person to swipe his fingerprint over the scanner, instead of pressing down upon it. As a finger's surface reaches thermal equilibrium with the sensor surface very quickly, it is difficult to get a good print from the thermal scanner using the touch method.

1.5 Fingerprint Classification

Fingerprints can be classified into several classes based on their ridge flow. As was stated in Section 1.3.3, we observe singularities in fingerprint structures which are caused by certain ridge flow patterns. For example, the loop is formed when ridges take a 180° turn, and a delta is formed when ridges meet at a Y-intersection. However, only a certain number of relative placements of these singularities have

been observed in fingerprints throughout history. Based on the presence of singularities, their placement relative to each other, and general ridge flow, fingerprints are divided into classes.

1.5.1 Classes

There are five primary classes into which fingerprints can be classified: left loops, right loops, whorls, arches and tented arches. Figure 1.12 shows examples of fingerprints from different classes.

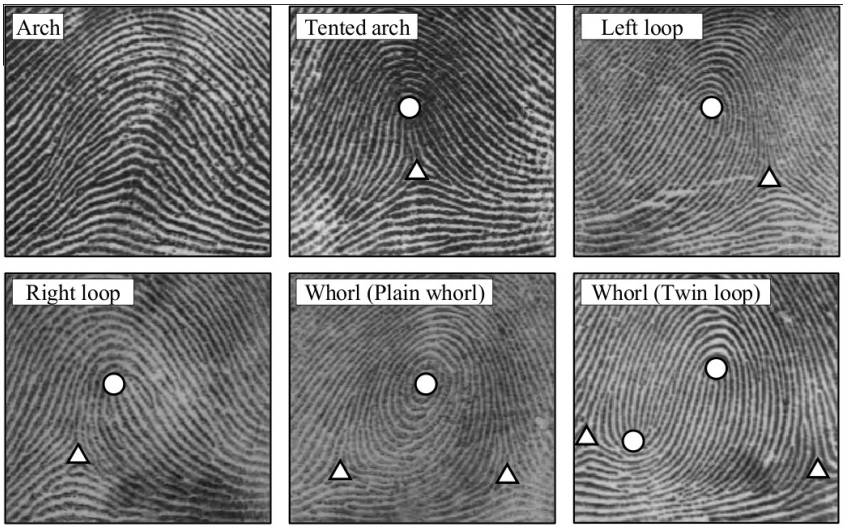

Figure 1.12 Fingerprints belonging to the five different classes. A circle denotes a loop, whereas a triangle denotes a delta. The number of singularities and their relative placement is the deciding factor in classification (taken from *Handbook of Fingerprint Recognition [47]*).

1.5.2 Classification Techniques

Classification of fingerprints into these classes is done based on several criteria. For instance, the positions of singularities present in the fingerprint image relative to each other is one way to classify the print. However, if the impression that is being classified isn't a rolled impression, then it is possible that there aren't any singularities present in the fingerprint. From Figure 1.12 we can observe that such a fingerprint would be classified into an arch category, which might be incorrect. Further, left and right loops might not show the presence of a delta for the same reason, even though they have one. Again, a classification based only on the singularities wouldn't conclusively determine the class of the fingerprint, as the relative position of the delta with respect to the core determines whether the print is a left loop or a right loop. Algorithms which use singularities to classify a fingerprint were proposed by Kawagoe and Tojo [35], Karu and Jain [33], Ratha *et al.* [61], Hong and Jain [25], Wang and Dai [74], and Zhang and Yan [85]. A better approach is to use the orientation field image or ridge flow to determine the class.

Based on a pattern in the ridge flow, we can distinguish a left loop from a right loop, even if there the presence of a delta isn't recorded.

Neural networks have also been used for the task of classification. They are of particular importance to this thesis, as they are amongst the first applications of neural networks to some fingerprint recognition task. Popular approaches that use neural networks for classification were proposed by Hughes and Green [28], Bowen [6], Wilson, Candela, and Watson [80] and Jain, Prabhakar, and Hong [29].

1.6 Fingerprint Matching

Fingerprint matching is the task of comparing two fingerprint images, and assigning a similarity score to the pair of images. This is a particularly hard task because of the extreme variations that might arise in different impressions of the same finger. The goal of a fingerprint matching algorithm is to assign a high similarity score to a pair of impressions of a finger disregarding properties like contrast, brightness, saturation, noise, size, translation, and rotation.

As a matching algorithm takes in two fingerprint images, one of them is a pre-recorded image, usually stored on a centralised server or database. The second image is a live-scan that needs to be authenticated. The pre-recorded image is called the *template*, and the live-scan is called the *target*, the *input*, or the *query*.

The large intra-class variability can stem from various factors: displacement, rotation, sensor type, finger pressure, moisture levels, presence of dirt, non-linear distortions, partial overlap, and noise. Further, errors in feature extraction can also affect fingerprint matching. To counter these factors, specific algorithms need to be employed. Various matching algorithms have been proposed, which can be classified into three categories: correlation-based, minutiae-based and non-minutiae based.

Correlation-based matching refers to assigning a score to two fingerprint images by computing the amount of overlap between the two images, after suitable alignment has been done. By alignment, we mean applying translations, rotations and scaling on one of the images, so that it aligns perfectly with the other image. Let $\mathbf{T}$ denote a template fingerprint image, and $\mathbf{I}$ denote an input image. Then, a correlation between the two images can be computed as

$$C(\mathbf{T}, \mathbf{I}) = \mathbf{T}^T\mathbf{I} \tag{1.5}$$

But this doesn't include any translations and rotations that might need to be applied to the input image. Call $\mathbf{I}^{(\Delta x, \Delta y, \theta)}$ the image obtained from $\mathbf{I}$, after applying a translations of $(\Delta x, \Delta y)$ and a rotation of θ on it. The similarity score is then given by

$$S(\mathbf{T}, \mathbf{I}) = \max_{\Delta x, \Delta y, \theta} C(\mathbf{T}, \mathbf{I}) \tag{1.6}$$

This simple method, however, doesn't tolerate intra-class variations that were mentioned earlier. Several improvements have been suggested to counter this. These include normalising correlation [15],

differential correlation [23], enhancing the fingerprints and thinning them before calculating the scores [38], and block processing [4, 83, 39].

Minutiae-based matching has several sub-approaches that work by extracting minutiae from a fingerprint image first. However, a general line of attack on the problem is aligning the extracted minutiae with each other. For this, let us define the problem more concretely:

We are given two sets of minutiae extracted from the template and the input fingerprint images. Let us call them M and M', respectively. Each minutia is typically represented by three values: the x-coordinate, the y-coordinate, and the orientation associated with the minutia. Hence, we have:

$$\begin{aligned} M &= \{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \ldots, \mathbf{m}_m\}; m_i = \langle x_i, y_i, \theta_i \rangle; 1 \leq i \leq m \\ M' &= \{\mathbf{m}'_1, \mathbf{m}'_2, \mathbf{m}'_3, \ldots, \mathbf{m}'_n\}; m'_j = \langle x'_j, y'_j, \theta'_j \rangle; 1 \leq j \leq n \end{aligned} \tag{1.7}$$

We need to determine a translation, rotation, and scaling of M', so that we have the highest similarity score between minutiae from M and M'. The similarity score is directly dependent on how closely two minutiae match. Further, two minutiae are said to *match*, if their spatial distance, and directional distance fall under a predefined threshold. So, $\mathbf{m}_i$ and $\mathbf{m}'_j$ match if:

$$SD\left(\mathbf{m}_i, \mathbf{m}'_j\right) = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2} \leq r_0; \text{and} \tag{1.8}$$

$$DD\left(\mathbf{m}_i, \mathbf{m}'_j\right) = \min\left(\left|\theta_i - \theta'_j\right|, 2\pi - \left|\theta_i - \theta'_j\right|\right) \leq \theta_0 \tag{1.9}$$

But we further require handling of non-linear transformations and scaling (in case the two fingerprints were taken by different sensors of different resolutions). There is vast amount of literature on this topic. However, as this isn't the focus of the thesis, we have decided to leave further exploration to the reader. Various algorithms proposed on minutiae-based matching can be found in [11, 30, 71, 78, 72, 9, 75, 61]

Non-minutiae-based matching algorithms are varied. Some of them use neural networks to determine whether two given fingerprints are a match or not [68, 58, 48]. Some methods propose the use of fingerprint curvature, rather than minutiae, to match them. If overlap between the template and input fingerprints isn't significant, minutiae based methods might fail to give a high score to a positive match. Matching on curvature, however, is feasible even in small overlaps (Yager and Amin, [82]). Another approach suggested by Parziale and Niel [57] suggests using Delaunay triangulation to achieve fingerprint matching.

1.7 Scope of the Thesis

In this thesis, we will talk about enhancement of fingerprint images using learning methods that are of unsupervised nature. These methods will focus on feature learning on the fingerprint images, or some property of the fingerprint images. Further, the enhancement of images shall be done using this learning, in that we will reconstruct fingerprint images based on what we learn from datasets of fingerprints.

Unsupervised feature learning is performed using continuous RBMs and convolutional deep belief networks, which have been briefly described in their respective chapters. The learning and enhancement processes have been described along with each algorithm. Each chapter has sections on experimental analysis, so as to compare results of existing algorithms with the proposed ones.

1.8 Outline of the Thesis

This thesis contains five chapters. Chapter 1, *Introduction*, serves as an introduction to biometrics, fingerprints, fingerprint features, and fingerprint recognition tasks. Fingerprint enhancement, which is the focus of this thesis, has been excluded from the introduction. Chapter 2, *Fingerprint Enhancement* serves as a literature review of existing enhancement algorithms. This chapter is more detailed than the introduction, and existing approaches are discussed more in-depth than their counterparts in Chapter 1. Chapter 1 presents a bird's-eye view of fingerprint recognition, and is essential to the rest of the thesis. Terminologies, conventions, details of fingerprints, and the like are detailed in this chapter.

Chapter 3, *Learning Local Fingerprint Orientations*, presents a contribution of the thesis. In this chapter, we explore how the task of fingerprint enhancement can be solved using machine learning and neural networks. We formulate the enhancement problem as learning the orientation field, after which contextual Gabor filtering can be used to give the enhanced image.

Chapter 4, *Hierarchical Learning of Fingerprint Features*, also presents a contribution of this thesis. This chapter, however, can be studied independently from Chapter 3, but we encourage the reader otherwise. Chapter 4 presents an approach to learn fingerprint structures directly from pixel images, without using previously extracted features. Further, we enhance fingerprint images using this learning at multiple resolutions.

Finally, Chapter 5, *Conclusions*, lists out the conclusions of the work included in this thesis. It highlights the importance and distinctiveness of the proposed work, and also outlines several possible paths for exploration that can be built upon the work included in Chapters 3 and 4.

Chapter 2

Fingerprint Enhancement

The performance of fingerprint identification systems is heavily dependent on the quality of fingerprint images that they are given as inputs. While good, clean fingers scanned using a clean sensor usually output images that can be directly used for feature extraction, the added process of fingerprint enhancement before recognizing them only adds to the accuracy of the system [26]. There are several ways in which distortions can be introduced in fingerprint images. Firstly, the sensors used to scan the fingerprints might be damaged themselves. A particular scenario is sensors that are used by the Government to mass-enroll citizens (for example, the recent introduction of *Aadhaar Card* required citizens to scan their fingerprints to store in a centralised system). During such enrolment drives, one scanner is used by thousands of people. Eventually, the scanner accumulates dirt, its parts might be damaged by improper use, or, if not cleaned properly, faded impressions from previously scanned fingers would still remain on the surface. All of these factors contribute towards noise in the fingerprint image. Secondly, the fingerprint to be scanned might itself contribute towards noise. If the person scanning his/her prints doesn't have clean hands, or have the right amount of moisture, his/her prints might get damaged. Further, a person's fingerprints might be degraded due his/her age or occupation (e.g., manual labourers, elderly people, people who have suffered burns). Their fingerprints are also usually not of a quality that allows recognition to be performed on them directly. Finally, fingerprints left at crime scenes have the most amount of noise. They are mostly clouded by large amounts of dirt, texture on the surfaces they were left on, or inter-mixed with other fingerprints. Such fingerprints, called *latent fingerprints*, are extremely important in forensic science, and their enhancement plays a crucial part in the identification process.

The reasons recounted above stress the need for fingerprint enhancement, and its help in fingerprint recognition and identification systems. In this chapter, we will form a background on the current work done in this area, which will help in a better understanding of the work presented in this thesis.

A fingerprint image has regions that can be classified into three types: well-defined, recoverable, and unrecoverable. Well-defined regions have clearly separable ridges and valleys, with consistent ridge flow. Recoverable regions might have small gaps, smudges, creases, etc,. but ridge structure can still be interpreted from them. Unrecoverable regions are so noisy that ridges are barely visible. With fingerprint

enhancement, we aim to improve consistency in ridge structure, make ridges and valleys distinguishable even in wet regions, and remove creases, in recoverable regions. Further, we also mark unrecoverable regions, so that they can be processed further. Some enhancement algorithms aim to also estimate the ridge structure in unrecoverable regions.

Figure 2.1 Fingerprint images of varying quality. From left: good quality; medium quality, distorted mostly by cuts; poor quality, varying moisture, cuts, and a lot of noise; latent fingerprint, taken from a crime scene, very little information.

2.1 Pixel-wise Enhancement and Median Filtering

Pixel-wise enhancement aims at enhancing individual pixel values so that some global parameters are satisfied. These algorithms generally introduce a greater disparity between values of pixels belonging to ridges and valleys. For example, Hong *et al.* [26] proposed a normalisation method in which pixel values are manipulated so as to attain a global mean and variance. Greenberg *et al.* [22] employ Weiner filtering in their approach. [37, 67] have also proposed adaptive normalisation techniques. Like Hong *et al.* , Kim and Park [37] use target parameters for mean and variance, except that they perform this operation block-wise, instead of setting them globally. Arpit and Namboodiri [3] proposed using Constrast-Limited Adaptive Histogram Equalisation (CLAHE) to stabilise the contrast in the input image before processing it. Another method to remove salt-and-pepper noise is to apply median filtering on the image. This removes small, pixel-sized, outliers. However, pixel-wise enhancement techniques don't include the essential information in the neighbourhood of a pixel. They, however, serve as good pre-processing steps for further enhancement.

2.2 Contextual Filtering

Contextual filtering is the most common method used for enhancement. By contextual filtering, we refer to filtering an image using separate filters for different pixels. The filter to be used at a point is determined by values of certain parameters at that point, that is, by the context of that point. Parameters that usually determine the filter are ridge orientation and ridge frequency. However, designing a separate

filter for every pixel is a time-consuming process. A workaround is to quantise different orientations and frequencies (thus reducing the unique number of orientations and frequencies), creating a filter corresponding to each pair of values, and then storing them in a bank. Filters from this bank can then directly be used when required.

In literature on enhancement, we will find numerous approaches to fingerprint enhancement that are based on contextual filtering. Some suggest filters that perform well for enhancement, while others improve existing algorithms by suggesting modifications to filtering and/or the way they are applied to the image. Overall, contextual filtering has the following two goals:

- *Low-pass filtering along the direction of the ridges.* This operation is aimed at connecting any discontinuities in ridge structure that might be present in the fingerprint. By rejecting high frequency data along the direction of the ridge, we effectively remove these discontinuities because they behave like edges.

- *Band-pass filtering perpendicular to the direction of ridge-flow.* This operation is aimed at increasing the distinctiveness of ridges and valleys. The frequencies that are allowed by the band-pass filter are determined by the frequency of ridges at that point. This removes data that inadvertently connects two ridges.

O'Gorman and Nickerson [51, 52] were the first to introduce contextual filtering for enhancement of fingerprint images. Their version involved generating a filter-bank from one filter, by orienting it in 16 directions. However, they didn't use ridge frequency as a parameter, and, instead, worked with minimum and maximum ridge and valley widths. Sherlock, Monroe, and Millard [66, 65] proposed filtering in the Fourier domain. They filter the input image with all filters from a filter-bank in the Fourier domain, which is equivalent to element-wise multiplication of the Fourier transforms of the image and the filters. Next, an inverse FFT is computed for the resultant images, and the best filter is chosen for each point, which determines the enhanced image. Hong, Wang, and Jain [26] proposed the use of Gabor filters for enhancement. This method became a widely-accepted standard for enhancement. The Gabor filter (Appendix A) is a two-dimensional Gaussian surface, that is multiplied along one direction by a cosine. The filter to be applied at a point is determined by the ridge orientation and ridge frequency. These two parameters also determine the direction the cosine will be oriented in, and its frequency. There are two more parameters used, which are variances along the ridge direction and perpendicular to it, of the Gaussian surface. A filter-bank is created using globally-observed values of these parameters, thus reducing computation. Yang *et al.* [84], Zhu, Yin and Yang [86], Erol, Halici and Ongun [18], Wu and Govindaraju [81], and Bernard *et al.* [5] have suggested approaches that build upon this using several modifications.

Figure 2.2 A fingerprint enhanced using Hong, Wang and Jain's [26], and Chikkerur, Cartwright, and Govindaraju's [14] algorithms. A binarisation is also performed on the enhanced image, by simply thresholding the enhanced image at its mean.

2.3 Frequency-domain Analysis

There have been approaches that work in the frequency-domain too, instead of the spatial domain, in while contextual filtering and pixel-wise enhancement work. Sherlock, Monroe, and Millard [66] proposed performing the filtering operation in the frequency domain, which is transforms convolution to element-wise multiplication. Watson, Candella and Grother [76], and Willis and Myers [79] proposed a block-processing algorithm which doesn't require computation of the orientation in the block. In this approach, the Fourier transform of the block is multiplied with a power of its power spectrum, and an inverse Fourier transform is computed, which gives the enhanced image:

$$I_{enh}[x,y] = F^{-1}\left(F\left(I[x,y]\right) \cdot \left|F\left(I[x,y]\right)\right|^{k}\right) \tag{2.1}$$

The power spectrum contains information on the ridge-flow direction. Figure 2.3 illustrates.

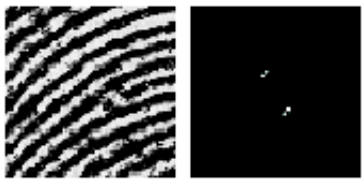

Figure 2.3 The power spectrum of the Fourier transform of a fingerprint patch. The original patch is on the left, and the power spectrum is shown as an image on the right. It can be seen that the peaks in the power spectrum, when joined, produce a line perpendicular to the direction of ridge-flow. This determines the orientation at that point.

Further work in enhancement in the Fourier domain, perhaps the most influential of them, was done by Chikkerur, Cartwright, and Govindaraju [14] in 2007. They used Short-time Fourier Transform (STFT) to determine local orientations and frequencies. These values were used to construct contextual filters in the frequency domain. They modelled the fingerprint structure in a block as a sine/cosine wave.

The idea was to use sliding window on a fingerprint image, and enhance each window separately. The window size was chosen so that orientation and frequency can be assumed to be constant in the window. Next, the orientation and frequency were calculated as a weighted average from the power spectrum of the Fourier transform.

2.4 Multi-Resolution Enhancement

Multi-resolution enhancement aims at analysing a fingerprint image at multiple frequency bands. Consider a fingerprint image at the scale of level 3 features (i.e., at individual ridges and pores). The irregularities that may arise due to presence of pores, or stray pixels, is high-frequency data. Further, consistent ridges and valleys are low-frequency data. Thus, appropriate processing on high-frequency data can smooth out ridges and valleys, thereby giving continuous ridges and valleys.

Algorithms based on multi-resolution analysis were proposed by Hsieh, Lai, and Wang [27], Cheng, Tian, and Zhang [12], and Almanasa and Lindeberg [1].

2.5 Formulating an Energy-Minimisation Problem

Fingerprint enhancement can also be looked at as an energy-minimisation problem. There have been recent developments on this too. The most promising one is enhancement of fingerprint images using Markov random fields (MRFs). Dass [16], Lee and Prabhakar [13] and Rama Reddy and Namboodiri [60], are some efforts towards enhancement using Markov random fields. Dass [16] suggested a Markov random field model for smoothing out orientation field over noisy regions and detecting singularities in images. Lee and Prabhakar [13] propose using a two-component MRF model, where the first component is a global mixture-model trained on two training images from each class of fingerprints, and the second component introduces a smoothness constraint over the orientation field. Their model was able to reduce the number of spurious minutiae considerably. Rama Reddy and Namboodiri [60] use a hierarchical Markov random field (HMRF). The HMRF has two stages (layers) that look at two different levels of the fingerprint. The enhancement process is iterative, and oscillates between these two layers. The input fingerprint is divided into overlapping patches, and for every patch, an ideal match is found from a set of existing patches. However, the best fit also depends on the four neighbouring patches. In very noisy regions, where the estimate of such a patch isn't possible, the iterative process ensures that the orientation field is estimated gradually.

Chapter 3

Learning Local Fingerprint Orientations

3.1 Introduction

Enhancement of fingerprint images plays an important part in fingerprint based authentication systems. This step plays a vital role so that noise from fingerprint images can be removed before they are subject to feature extraction or matching, and hence, decrease the probability of a false match. Noise in fingerprints can appear because of several reasons including, but not limited to, sensor noise, damp finger at the time of recording of the fingerprint, bruises and cuts in fingers, and non-uniform finger pressure [47]. Several approaches to fingerprint enhancement have been proposed in the past. Some of these approaches try to model the noise in the fingerprint image and remove it. Such methods include median filtering, contrast limited adaptive histogram equalisation, and Wiener filtering [22]. These techniques, however, are not very successful in modelling the kind of noise present in the fingerprint image for it can be a mixture of several kinds of noise. These techniques, however, aren't completely unused, and serve as pre-processing steps for more complicated enhancement algorithms.

Neural network approaches to pattern recognition and classification tasks had fallen out of favour in the 1990s after the advent of support vector machines (SVMs). However, they started gaining popularity again in the mid-2000s as new methods to train them surfaced [8]. The use of many-layered neural networks to learn patterns gave rise to the term "deep learning". Deep learning can be achieved using a variety of models. Deep belief nets, for instance, are examples of deep networks. Deep belief nets are built by stacking several restricted Boltzmann machines (RBMs) over each other. The RBMs, as independent models, have been used as generative models for several years [70, 63, 62, 59] with the problems addressed by them [24] spanning from bag-of-words for document representation [62], to user ratings of movies [63], and modelling video [70], and speech [59]. RBMs have a visible layer and a hidden layer of neurons, with every neuron in the visible layer connected to every neuron in the hidden layer, and there being no intra-layer connections. Traditionally, however, RBMs work only on binary inputs. Chen and Murray [10] proposed a model which was a variant of the RBM, and can work on continuous inputs. The neurons in the model proposed by Chen and Murray (a Continuous RBM), are continuous stochastic units formed by adding a zero-mean Gaussian noise to the total input received

by them. We shall use this model in this chapter to encode and learn orientation fields of fingerprint images.

3.2 Related Work: A Recall

Several techniques currently used in fingerprint enhancement involve contextual filtering - a primary filter used being the Gabor filter. The use of Gabor filter for fingerprint enhancement was proposed by Hong, *et al.* [26]. Parameters for the filters used on the image are determined by local ridge orientation, local ridge frequency and reliability of ridge orientation. Yet earlier attempts involved the use of elongated filters in the direction of ridges [53]. Unlike these, methods that operate in the frequency domain [66] and those which work in both domains [14] have also been attempted. The algorithm proposed by Hong *et al.* involves a bank of Gabor filters that have optimal joint resolution in both spatial and frequency domains. This method has undergone several changes, with major improvements being proposed by Greenberg *et al.* [22], Bernard *et al.* [5], Yang *et al.* [84] and Zhu *et al.* [86]. A recent effort by Sutthiwichaiporn *et al.* [69], enhances fingerprints using a spatial-frequency domain approach with matched filtering and diffusion of high quality enhanced zones into low quality zones in an iterative-feedback manner. Lee *et al.* [13] incorporated steerable filters with Markov random fields for enhancement.

In this chapter, we attempt to learn various types of local orientation field patterns observed in fingerprints by training neural networks. We used two continuous restricted Boltzmann machines for this task. They were trained in an unsupervised manner initially, and then a pass of backpropagation learning was introduced to further correct the weight matrices.

In this chapter, we explore an approach in which we learn orientation fields over local regions, of size 60 pixels $\times 60$ pixels, and use this learning to reconstruct distorted fields observed in noisy images. This tries to formulate the problem as learning of ideal orientation fields instead of the noise, and removing the noise. This is unlike other methods which try to model the noise observed in fingerprints, and try to remove it. To achieve this, we look at the restricted Boltzmann machine as the required model. An RBM can be trained on several images to learn patterns present in them. This is an example of unsupervised feature learning. Furthermore, this trained RBM, when shown a random input, will try to reconstruct it to reflect the patterns it has already learnt.

Figure 3.1 gives an example of how an RBM learnt on images of hand-written 2s sticks to features that it has extracted from these images, and tries to see all input images as instance of other 2s. The reconstruction of the image of the hand-written 3 is closer to a 2 than it is to a 3.

This property enjoyed by restricted Boltzmann machines prompted us to explore its application in this task. The proposed hypothesis is as follows: given that a trained RBM can reconstruct input images to look like ones it was trained on, we expect an RBM trained on local and ideal orientation fields to remove noise, if present, from an input orientation field.

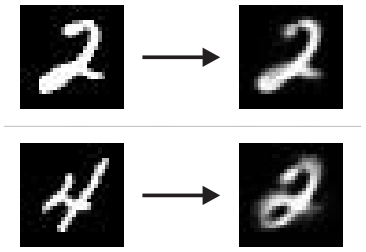

Figure 3.1 Reconstructions performed by an RBM which was learnt on images of the hand-written instances of the digit "2".

3.3 Restricted Boltzmann Machines

Before we move to the proposed approach, it is important to recall the restricted Boltzmann machine. A restricted Boltzmann machine (RBM) is a special form of an energy-based model. It has two layers of units - visible or input, and hidden. Each visible neuron is connected to every hidden neuron via. a weighted and undirected edge. This weight denotes the weighing factor that is applied to any signals that are sent from this visible neuron to the hidden neuron, or vice versa. The graph thus formed, by considering every neuron as a vertex and every weight as an edge, has "restrictions" on it This graph must be a bipartite graph, with the visible neurons and the hidden neurons forming two disjoint sets. This means that there shall be no weights between visible neurons, as shall there not be weights between two hidden neurons. A graphical representation of the RBM is given in Figure 3.2.

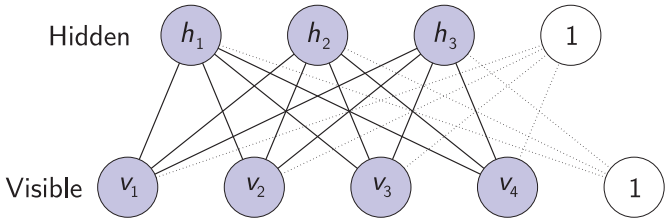

Figure 3.2 A graphical representation of the restricted Boltzmann Machines. There are two sets of neurons: visible and hidden, with connections between every pair of visible and hidden neurons. Connections are weighted and undirected, which is why this network isn't a feed-forward neural network. Solid connections denote weights between visible and hidden neurons. Non-coloured neurons are bias units. Values of these units are always set to 1. The connections between them and a neuron denote the bias for that neuron.

Let us denote the number of visible units (neurons) by V, and the number of hidden units by H. Each unit has a bias term associated with it, which is an integral part of the learning. At each update of the parameters, the bias terms are also updated along with the weights. Bias terms play an important part in learning representations. Let us call the set of biases for visible units (or "visible biases") by the vector $\mathbf{b} = [b_1, b_2, \ldots, b_V]^T$. Similarly, the hidden biases are $\mathbf{c} = [c_1, c_2, \ldots, c_H]^T$. Denote by w_{ji}, the weight between the i-th visible unit and the j-th hidden unit. We arrange these weights into an $H \times V$

matrix to get the weight matrix, $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1i} & \dots & w_{1V} \\ w_{21} & w_{22} & \dots & w_{2i} & \dots & w_{2V} \\ \vdots & \vdots & & \ddots & & \vdots \\ w_{H1} & w_{H2} & \dots & w_{Hi} & \dots & w_{HV} \end{bmatrix} \tag{3.1}$$

As the RBM is an energy-based probabilistic model, there is a probability associated with every configuration of the variables of interest. To define this probability, we have to first define the energy associated with a configuration of the network. This energy is dictated by the hidden and visible units, and the weights and biases. The energy of the RBM for a certain set of visible and hidden units is given by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \tag{3.2}$$

where $\mathbf{v} = [v_1, v_2, \dots, v_V]^T$ and $\mathbf{h} = [h_1, h_2, \dots, h_H]^T$ are activations of the visible and hidden units, respectively. This model is ready for us to train now. If we look at the energy function in terms of a function of the weights and biases (each of which are an independent dimension), we can imagine it to be a surface in a very high-dimensional space. Any configuration of the network can be represented as a point on this surface, and it is possible to move on it by applying updates or changes to the parameters. As the dimensions are independent of each other, changes with respect to one parameter aren't affected by any other parameter. The training process starts with a random configuration of the network (random values for the weights and biases), which can be denoted as a point on this surface, and aims to lead the system to a configuration which will have as low an energy as possible. It follows, then, that we need an update rule which will reduce the energy of the system after every epoch, i.e., every iteration of training. It is the weights and biases, that we had initialised randomly, that the training algorithm applies updates to after every epoch. To calculate these updates, we differentiate the energy function with respect to each of these quantities, and then follow a gradient descent update, which will add a quantity to the weights and biases so that the new configuration moves in a direction exactly opposite to the direction of maximum gradient.

Consider the weight between the i-th visible neuron and the j-th hidden neuron - w_{ji}. Given this state of the network, we have certain configurations of the visible and hidden units: $\mathbf{v}$ and $\mathbf{h}$. To calculate the change to weight w_{ji}, we differentiate the energy function with respect to w_{ji}. From Equation 3.2, we have

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= -\mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \\ &= -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} h_j w_{ji} v_i \end{aligned} \tag{3.3}$$

Differentiating with respect to w_{ji},

$$\begin{aligned}
\frac{\partial E}{\partial w_{ji}} &= -\frac{\partial}{\partial w_{ji}} \left(\sum_{i'} b_{i'} v_{i'} - \sum_{j'} c_{j'} h_{j'} - \sum_{i',j'} h_{j'} w_{j'i'} v_{i'} \right) \\
&= -\frac{\partial}{\partial w_{ji}} \left(\sum_{i',j'} h_{j'} w_{j'i'} v_{i'} \right) \\
&= -h_j v_i - \sum_{i' \neq i, j' \neq j} h_{j'} w_{j'i'} v_{i'} \\
&= -h_j v_i
\end{aligned} \tag{3.4}$$

Because the parameters are independent of each other, derivatives of the biases with respect to w_{ji} are zero, and so are derivatives of $w_{j'i'}, (i' \neq i, j' \neq j)$. The update to weight w_{ji} is hence dependent on h_j and v_i.

3.3.1 Alternate Gibbs Sampling and Contrastive Divergence Learning

We have established so far, the structure of the RBM. We know that the RBM is a bipartite graph. Hence, the visible units are independent of each other, and so are the hidden units. This is a good advantage for calculating probabilities of visible units given a set of hidden units, and vice versa, and can be done in an alternate manner. This can be easily visualised as follows: $\mathbf{v}^0 \rightarrow \mathbf{h}^0 \rightarrow \mathbf{v}^1 \rightarrow \mathbf{h}^1 \ldots$ This process of calculating probabilities and sampling from subsequent distributions in a back-and-forth manner is called alternate Gibbs sampling. To calculate activations of neurons at each step of this path, we need to write equations for probabilities $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$. It is found that an update rule given in Equation 3.5 is a well fit for training an RBM [8].

$$\begin{aligned}
\Delta w_{ji} &\propto \left(v_i^0 h_j^0 - v_i^\infty h_j^\infty \right) \\
&= \eta \left(v_i^0 h_j^0 - v_i^\infty h_j^\infty \right)
\end{aligned} \tag{3.5}$$

However, to perform an update using this training algorithm would mean performing alternate sampling forever, which isn't feasible when considering real-world algorithms. Hinton showed [8] that performing the sampling for only K steps, instead of going to infinity, approximates the original objective function closely. Even though this introduces a bias in the learning, this bias doesn't affect our model much. This modifies our learning algorithm as follows:

$$\Delta w_{ji} = \eta \left(v_i^0 h_j^0 - v_i^K h_j^K \right) \tag{3.6}$$

Algorithm 1 Training Algorithm for an RBM

Input: Training Images, $\mathbf{T}$
$N_T \leftarrow$ size($\mathbf{T}$)
Initialise W with normally distributed random numbers
repeat
  $\Delta W, \Delta b, \Delta c \leftarrow 0$
  for $m = 1 \rightarrow N_T$ **do**
    $\mathbf{v} \leftarrow \mathbf{T}(m)$ $\{\mathbf{T}(m)$ is the m-th image$\}$
    Compute $\mathbf{h}$ from $\mathbf{v}$ and $\mathbf{c}$
    for $k = 1 \rightarrow K$ **do**
      Compute $\mathbf{v}$ from $\mathbf{h}$ and $\mathbf{b}$
      Compute $\mathbf{h}$ from $\mathbf{v}$ and $\mathbf{c}$
    $\Delta W \leftarrow \Delta W + \eta \left((\mathbf{v}^0)^T \mathbf{h}^0 - (\mathbf{v}^K)^T \mathbf{h}^K\right)$
    $\Delta b \leftarrow \Delta b + \eta \left(\mathbf{v}^0 - \mathbf{v}^K\right)$
    $\Delta c \leftarrow \Delta c + \eta \left(\mathbf{h}^0 - \mathbf{h}^K\right)$
  $W \leftarrow W + \frac{1}{N_T} \Delta W$
  $b \leftarrow b + \frac{1}{N_T} \Delta b$
  $c \leftarrow c + \frac{1}{N_T} \Delta c$
until convergence

3.3.2 Calculating Activations of Neurons

The states of neurons are derived from their activations. The activation of a neuron is determined by the total input received by the neuron from all neurons connecting to it. For instance, for a visible neuron, the total input is calculated as the weighted sum of states of all hidden neurons, with the weights being the weights of the connections. Consider a hidden neuron j. The total input received by j is:

$$X_j = c_j + \sum_{i=1}^{V} w_{ji} v_i \tag{3.7}$$

This input is passed through a sigmoid function to give us the probability that it is turned on. Hence,

$$P(h_j = 1|\mathbf{v}) = \sigma(X_j) = \frac{1}{1 + \exp(-X_j)} = \frac{\exp\left(c_j + \sum_{i=1}^{V} w_{ji} v_i\right)}{1 + \exp\left(c_j + \sum_{i=1}^{V} w_{ji} v_i\right)} \tag{3.8}$$

Once we have calculated the probability of being on for every hidden neuron, we sample from these probabilities to get states of the hidden neurons. This is simple binomial sampling, where there are two possible outcomes with probabilities p and $1 - p$. Sampling is done by drawing a random sample from a uniform distribution, and choosing ON or OFF depending on whether this chosen value is greater than p or less than p. The activations of visible neurons are calculated from hidden neurons in a similar manner. The entire training algorithm for an RBM is summarised in Algorithm 1.

3.4 Proposed Approach

The proposed approach is divided into three stages: generating orientation fields of query images using a gradient-based method; training two continuous restricted Boltzmann machines (CRBMs) on a database of orientation field images; and feeding the orientation fields of query images to these RBMs and using the outputs for Gabor filtering. We will describe each stage here.

3.4.1 Gradient-based approach to estimate orientation field

We use the algorithm given by Hong *et al.* in [26] for an initial estimate of the orientation field. This algorithm performs very well in noiseless images, but doesn't give robust estimates of the orientation field for noisy images as it considers a small region during estimation. The process of obtaining this orientation field is summarised here:

1. Divide the given image into blocks of sizes $W \times W$. Consider a block centred at a pixel (i, j). Compute the horizontal and vertical gradients using Sobel operators to obtain $\partial_x(i, j)$ and $\partial_y(i, j)$, respectively.

2. Obtain an estimate of the orientation field, $\theta(i, j)$ using the following equations:

$$\mathcal{V}_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2\partial_x(u,v)\partial_y(u,v) \tag{3.9}$$

$$\mathcal{V}_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} \partial_x(u,v)^2\partial_y(u,v)^2 \tag{3.10}$$

$$\theta(i,j) = \frac{1}{2}\arctan\frac{\mathcal{V}_y(i,j)}{\mathcal{V}_x(i,j)} \tag{3.11}$$

3. Convert the orientation field into a continuous vector field:

$$\Phi_x(i,j) = \cos(2\theta(i,j)) \tag{3.12}$$

$$\Phi_y(i,j) = \sin(2\theta(i,j)) \tag{3.13}$$

Perform Gaussian smoothing to get Φ'_x and Φ'_y:

$$\Phi'_x(i,j) = \sum_{u=\frac{-w_\phi}{2}}^{\frac{w_\phi}{2}} \sum_{v=\frac{-w_\phi}{2}}^{\frac{w_\phi}{2}} G(u,v)\Phi_x(i-uw, j-uw) \tag{3.14}$$

$$\Phi'_y(i,j) = \sum_{u=\frac{-w_\phi}{2}}^{\frac{w_\phi}{2}} \sum_{v=\frac{-w_\phi}{2}}^{\frac{w_\phi}{2}} G(u,v)\Phi_y(i-uw, j-uw) \tag{3.15}$$

where G is a Gaussian low-pass filter of size $w_\phi \times w_\phi$. The orientation field is now obtained as:

$$O(i,j) = \frac{1}{2} \arctan \frac{\Phi'_y(i,j)}{\Phi'_x(i,j)} \tag{3.16}$$

The orientation field calculated using this approach is used as input in the next step. It is important to note that the orientation field values generated by this algorithm lie in the range $[0, \pi]$, as an orientation of α and $\pi + \alpha$ is essentially the same.

3.4.2 Training the CRBMs

The focus of this approach is training of the continuous restricted Boltzmann machines. The RBMs are trained using noiseless orientation fields so that they learn fingerprint ridge patterns. The training images used in this step were obtained by orientation field estimation of fingerprint patches taken from enhanced images. It was ensured that these enhanced images did not have noise, so that the training samples were noiseless.

The Model: The classical RBM takes only binary inputs, so we used a variant of the same - the continuous restricted Boltzmann machine (CRBM) [10] - as our model. The CRBM inputs are normalised to $[0, 1]$. We train two CRBMs, hence the significance of 0 and 1 for both of them is different. To ensure that we get continuity in our output, we don't use the orientation value directly for training. To emphasise this, consider the orientation field values near the apex of an upward-curved ridge. Approaching the apex from left, the orientation tends to π, whereas approaching it from the right, it tends to 0. It is due to this inconsistency that we do not use orientation values for training. Training a CRBM using the orientation field, θ, gives rise to anomalies in the resulting image when the orientation has to jump from π to 0. This gives rise to inconsistent outputs of the CRBM, wherein it doesn't jump directly from π to 0. To counter this, we break down θ into two functions, $s(\theta)$ and $c(\theta)$ as follows:

$$s(x) = \begin{cases} \frac{4}{\pi}x & \text{for } 0 \leq x < \frac{\pi}{4}; \\ -\frac{4}{\pi}x + 2 & \text{for } \frac{\pi}{4} \leq x < \frac{3\pi}{4}; \\ \frac{4}{\pi}x - 4 & \text{for } \frac{3\pi}{4} \leq x \leq \pi \end{cases} \tag{3.17}$$

$$c(x) = \begin{cases} -\frac{4}{\pi}x + 1 & \text{for } 0 \leq x < \frac{\pi}{2}; \\ \frac{4}{\pi}x - 3 & \text{for } \frac{\pi}{2} \leq x \leq \pi \end{cases} \tag{3.18}$$

Both $s(\theta)$ and $c(\theta)$ are continuous functions even when the range $[0, \pi]$ is looped, i.e., $s(0) = s(\pi)$ and $c(0) = c(\pi)$. $s(\theta)$ and $s(\theta)$ together are enough to encode θ. One CRBM (referred to as "c-CRBM" here onwards) is trained with the images obtained by applying $c(x)$ on the orientation angles, and other CRBM (referred to as "s-CRBM" here onwards) with those formed by applying $s(x)$. Figure 3.3 shows the two functions $s(x)$ and $c(x)$.

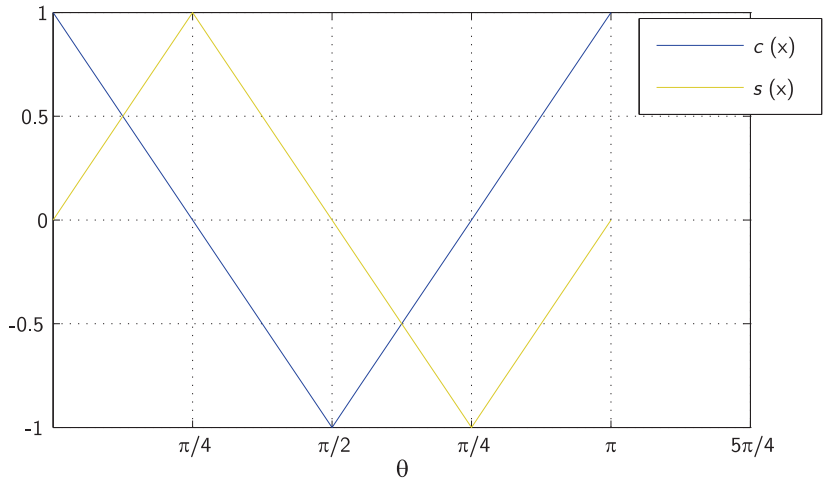

Figure 3.3 The functions $s(x)$ and $c(x)$ used to encode θ. Both $s(x)$ and $c(x)$ are continuous over $[0, \pi]$, with $s(x + \pi) = s(x)$ and $c(x + \pi) = c(x)$. These functions are applied on the orientation field images to generate two images which are then fed to the s-CRBM and c-CRBM, respectively.

Figure 3.4 Training images for the c-CRBM (top) and the s-CRBM (bottom). These images were generated after applying the functions $s(x)$ and $c(x)$ (Figure 3.3) to 60 pixels $\times$60 pixels-sized orientation field images extracted from Gabor-enhanced fingerprints.

Figure 3.4 gives examples of training images used for the c-CRBM and the s-CRBM. Each orientation field patch, of size 60 pixels $\times$60 pixels, extracted from a Gabor-enhanced fingerprint gave two images after applying $s(x)$ and $c(x)$.

Parameters and training: As described in [10], a CRBM is different from the traditional RBM. The CRBM has two layers - input (or visible) and hidden - wherein every neuron in the input layer is connected to every neuron in the hidden layer. These are undirected, weighted edges, hence the hidden layer can feed inputs to the visible layer as well. A neuron in either layer emulates a sigmoid function. This sigmoid is applied on the total input received at that neuron plus a zero-mean Gaussian noise. Let j be a neuron in the hidden layer, and let $\mathbf{v} = [v_1, v_2, \ldots, v_V]^T$ be the vector denoting the values at neurons at the visible layer. Let $\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jV}]^T$ be the set of weights corresponding to the j-th neuron in the hidden layer. An additional parameter is introduced for every neuron in a layer for the CRBM. This parameter, called the "noise-control" parameter and denoted by a_j, controls the nature

of the neuron's stochastic behaviour. The output of the j-th neuron in the hidden layer is then given by

$$s_j = \phi(\theta_j) = \frac{1}{1+\exp(-\theta_j)}; \text{where } \theta_j = a_j x_j \tag{3.19}$$

and x_j denotes the total input at neuron j:

$$x_j = \sum_{i=1}^{V} w_{ji} v_i + b_j + \sigma N(0,1) \tag{3.20}$$

σ and $N(0,1)$ make up the noise component, where σ is a constant for the whole network, and $N(0,1)$ is a Gaussian random variable with zero mean and unit variance. Both $\mathbf{w}_j$ and a_j are updated for every neuron j after each epoch. The quantity b_j is the bias added to every unit (Figure 3.5). The update rules are given by

$$\Delta w_{ji} = \eta_w \left(\langle v_i s_j \rangle - \langle \hat{v}_i \hat{s}_j \rangle \right) \tag{3.21}$$

$$\Delta a_j = \frac{\eta_a}{a_j^2} \left(\langle s_j^2 \rangle - \langle \hat{s}_j^2 \rangle \right) \tag{3.22}$$

where $\hat{v}_i$, $\hat{s}_j$ denote the one-step sampled state of neurons i and j, respectively; η_w and η_a are learning rates; and $\langle \cdot \rangle$ denotes the expected value over all training samples. The learning rates for the c-CRBM used in training were $\eta_w = 5$ and $\eta_a = 0.5$, while those used for the s-CRBM were $\eta_w = 4$ and $\eta_a = 0.5$. The set of weights used to calculate activations of hidden neurons from the visible neurons is the transpose of the set of weights used to calculate the activations of the visible neurons from the hidden neurons.

In this work, we focused on training with fingerprint patches of sizes 60 pixels $\times$ 60 pixels. However, exploiting the similarity in orientation field observed in pixels which are neighbours, we used only 100 neurons in the visible layer of CRBMs. This was done by first resizing the training patches to 10×10, and then assigning each pixel in the resulting patch a unique neuron in the visible layer. In the hidden layer, we used 90 neurons. Training was performed using 4580 training samples, with 25000 epochs per CRBM.

The CRBMs were trained in an unsupervised manner according to these training rules, and were then subject to a pass of backpropagation learning to further correct the weights. Before this pass, the CRBMs are "opened" up to get an output layer independent of an input layer. In this process, the symmetric nature of weights in the pre-training phase is removed, and both set of weights become independent of one another. The backpropagation algorithm for the CRBM is different from the one for an RBM because of parameters not included in the RBM. The updates for backpropagation learning hence needed to be deduced. These have been laid out here. Call the number of layers in the opened up network, L (starting from 0). Here, $L = 2$ for the CRBMs. Let y_i denote the expected/ideal value of the output at neuron i in the output layer. Let X_i^l denote the output of the i-th neuron in layer l, w_{ij}^l denote the weight between i-th neuron in layer l and j-th neuron in layer $l-1$; n_l denote the number of neurons in layer l; and a_i^l denote the noise-control parameter at the i-th neuron in layer l. The backward pass equation for layer L is as follows:

$$\delta_i^L = (y_i - X_i^L) \cdot a_i^L \cdot \phi'(\theta_i^L) \tag{3.23}$$

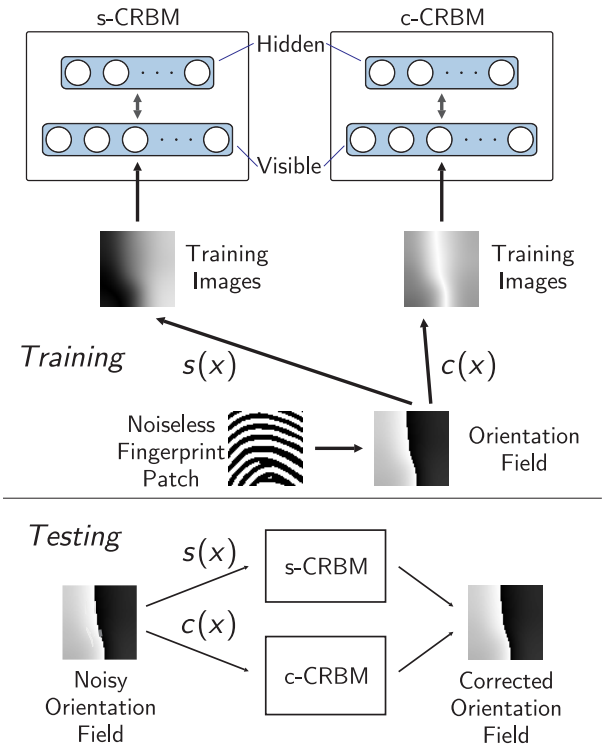

Figure 3.5 The model, training and testing processes. *top (training)*: A representation of the continuous RBM used in this work. The first two steps of the proposed approach are summarised. *bottom (testing)*: A representation of the third step in the proposed approach.

where $\phi'(x) = \phi(x)(1 - \phi(x))$ is the derivative of the sigmoid function, and θ_i^L is the product of the noise-control parameter and the total input at neuron i given by

$$\theta_i^L = a_i^L \left(\sum_{j=1}^{n_{L-1}} w_{ij}^L X_j^{L-1} + b_j + \sigma N(0,1) \right) \tag{3.24}$$

Backward pass in the remaining layers is given by:

$$\delta_i^l = \phi'(\theta_i^l) \cdot a_i^l \cdot \sum_{t=1}^{n_{l+1}} \delta_t^{l+1} w_{ti}^{l+1}; \text{where } 0 < l < L \tag{3.25}$$

Equations 3.25 and 3.23 lead to the following update rules for $\mathbf{w}_j$ and a_j:

$$\Delta w_{ij}^l = \nu_w \cdot \delta_i^l \cdot X_j^{l-1} \tag{3.26}$$

$$\Delta a_i^l = \nu_a \cdot \frac{\delta_i^l}{a_i^l} \cdot x_j^l = \nu_a \cdot \frac{\delta_i^l}{(a_i^l)^2} \cdot \theta_i^l \tag{3.27}$$

where $x_j^l = \left(\sum_{j=1}^{n_{l-1}} w_{ij}^l X_j^{l-1} + b_j + \sigma N(0,1) \right)$ is the total input received at the j-th neuron in layer l, and ν_w and ν_a are learning rates for the weights and the noise-control parameter, respectively. In our

experiments, we set the learning rates for the weights and the noise control parameter to be equal. The values used were $\nu_w = \nu_a = 1$ for the c-CRBM, and $\nu_w = \nu_a = 0.9$ for the s-CRBM.

Both CRBMs are trained with backpropagation learning according to these update rules for 50000 epochs.

3.4.3 Gabor filtering using outputs of the CRBM

Once the CRBMs have been trained, they are fed inputs from a database. We use the FVC 2002 Db3_a [46] database to run our tests. Before feeding these images to the CRBMs, their orientation fields are calculated using the algorithm mentioned in section 3.4.1. Then two images are generated for every image in the database by applying the functions $s(x)$ and $c(x)$ on the orientation fields. These images are then fed to the respective CRBMs and their outputs recorded. Now, using the two output images for every image in the database, we can generate the orientation field for that image. Let cIm and sIm denote, respectively, the outputs for the c-CRBM and the s-CRBM for an image Im. To get the orientation field using these matrices, the following steps are performed:

1. A matrix oIm with the same dimensions and size as cIm and sIm is created.

2. For all pixels in sIm whose value is greater than or equal to 0.5, the corresponding pixel in oIm is assigned a pixel value of $\frac{\pi}{4}(1 - c)$, where c is the value at the corresponding pixel in cIm.

3. The same is done for pixels in sIm with values less than 0.5, except that a value of $\frac{\pi}{4}(3 + c)$ is assigned instead.

4. The matrix oIm now has the desired orientation field.

The above sequence of steps generates an orientation field from the s- and c-CRBM outputs, which can directly be used for Gabor filtering. Instead of orientation generated using our gradient-based method (section 3.4.1), we use the orientation field generated by the CRBMs. The frequency image can directly be obtained from the original image. The frequency and orientation images together dictate the natures of the Gabor filters, which are then used appropriately to filter and enhance images from the database.

3.5 Experimental Results

We show, on the basis of results of experiments conducted using the above set-up, that this model can correct orientation fields in fingerprints where noise is present in small-sized, local regions.

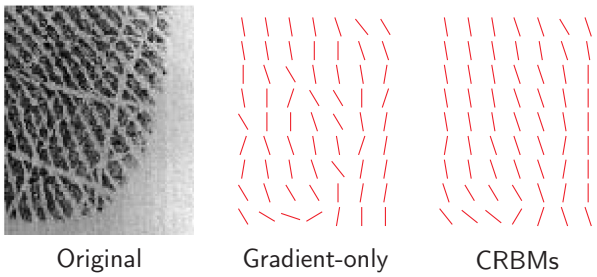

Figure 3.6 Correction performed in the orientation field by the Continuous RBMs. Orientation fields in the center of the fingerprint patch were distorted by creases. These have been corrected by the CRBMs.

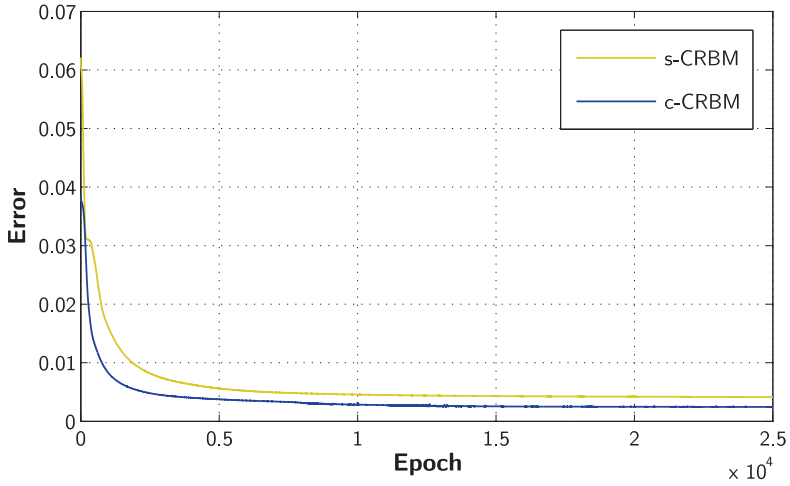

Figure 3.7 A plot of error progression for each of the two Continuous RBMs plotted versus epochs in training.

3.5.1 Experimental Set-up

The Continuous RBM required for this task was programmed in Python 2.6.5 [GCC 4.4.3]. The program was setup to work on an Nvidia CUDA GeForce GTX 580 graphics processing unit with 512 cores and 1 GB of memory. The cudamat [49] library for python was used to call GPU operations. Unsupervised pre-training for 25000 epochs took 88 minutes per CRBM and backpropagation training for 50000 epochs took 83 minutes. The error progression during unsupervised pre-training is shown in Figure 3.7.

3.5.2 Weights

It is possible to visualise the weights learnt by the continuous RBMs. Recall that we have 100 neurons in the visible layer and 90 neurons in the hidden layer. As all visible neurons are connected to all hidden neurons, we have 100 weighted connections for every hidden neuron. To visualise all the weights of the continuous RBM, we partition the set of connections into 90 sets - one for each hidden

Figure 3.8 A comparison between Gradient-only orientation field estimation and the proposed approach. The proposed approach is more successful in removing creases, and doesn't distort the rest of the orientation field estimated by the gradient-based method.

neuron. In the j-th set, we place all the weights corresponding to the j-th hidden neuron. Each set of weights can now be converted into an image for visualisation. This is done by re-stacking the 100 weights associated with a visible neuron into a 2-D matrix. The order of stacking is dictated by the order in which training images were flattened into an array to be fed to the visible units. So, if images were flattened in a row-first manner, the re-stacking is done in the same way. In our implementation we used the row-first flattening rule.

Using this technique, we visualised the weights learnt by both the networks. Figures 3.9 and 3.10 are visualisations of weights of the c-CRBM and the s-CRBM, respectively.

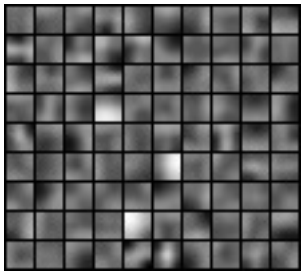

Figure 3.9 Visualising the weights of the cosine CRBM after unsupervised pre-training. The weights are flattened in the weight matrix, and have been stacked in a row-first manner for this visualisation.

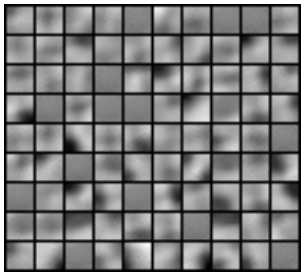

Figure 3.10 Visualising the weights of the sine CRBM after unsupervised pre-training. The weights are flattened in the weight matrix, and have been stacked in a row-first manner for this visualisation.

3.5.3 Qualitative Analysis

We perform a qualitative analysis with the gradient-based orientation field enhancement algorithm. We used the code provided by Peter Kovesi [40] to test the gradient-based algorithm against ours. We have tested our algorithm on all fingerprints from the Db3_a database of the FVC 2002 set of databases. It was observed that this algorithm improves on this orientation field generated by the gradient-based algorithm in images where noise is present in small regions and which causes small, local discontinuities in the ridge structure. Further, it was observed that regions where the gradient method performed well weren't distorted by the outputs of the CRBMs. The proposed approach also performs better than gradient-only coupled with Gabor in removing creases from fingerprints. Figure 3.8 illustrates an example.

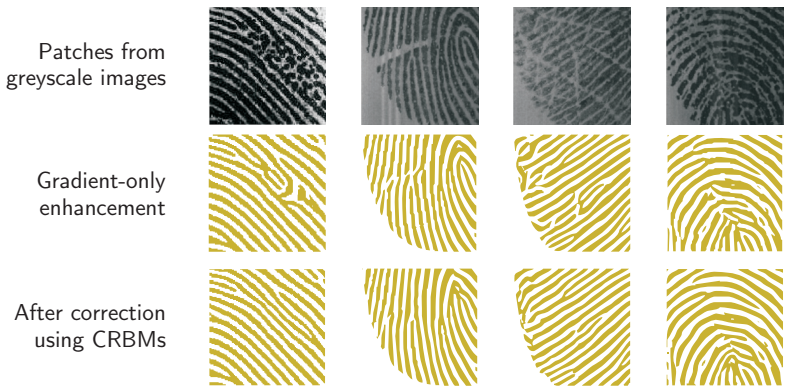

Figure 3.11 Examples of corrections in orientation fields and the resulting enhancements on noisy patches from some fingerprints. It can be seen that the corrections performed by the CRBMs remove local discontinuities in ridge structures. Further, they do not disrupt ridge structures that weren't noisy or distorted, thus indicating that the learning of local orientation fields was sound.

3.5.4 Quantitative Analysis

Quantitative analysis was performed on this algorithm by plotting the false accept and genuine accept rates output by the NIST matcher, bozorth3 [50], when fingerprints enhanced with the proposed approach were given as input. For comparison, we have also shown the plot of false accept and genuine accept rates corresponding to the gradient-based Gabor enhancement of fingerprints. Figure 3.12 displays these ROC curves. The database of fingerprints used was again Db3_a of the FVC 2002 set of databases. ROC curves have been plotted after performing 2800 genuine comparisons, and 4950 impostor comparisons.

It can be seen from the ROC curves that the proposed algorithm performs better than just gradient-based Gabor enhancement.

We also calculated the number of spurious and missing minutiae generated by our algorithm on the FVC2002 Db3_a database, using the ground truth data provided by Umut Uludag [36]. We observed

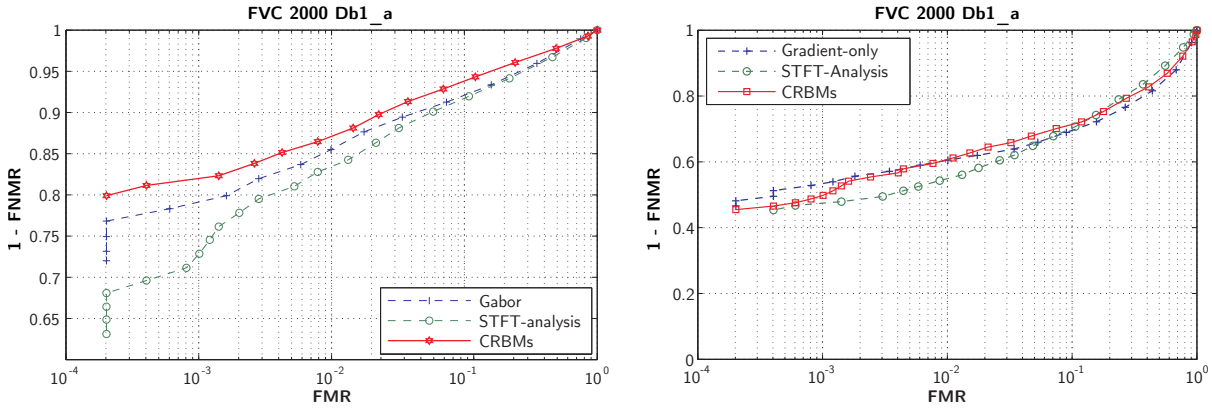

Figure 3.12 ROC Curves: Genuine Accept Rate plotted vs False Accept Rate. It is seen that the proposed approach is an improvement over gradient-only.

that CRBMs reduce the number of spurious and missing minutiae detected by the gradient-only method. We report in table 3.1, the spurious and missing minutiae count:

Method	Ground Truth	Total detected	Spurious	Missing	EER
Gradient-only	**19032**	53389	38415 (71.95%)	4058 (21.32%)	24.34
STFT-analysis		48963	33096 (67.59%)	**3165 (16.63%)**	21.99
CRBMs		**41764**	**26324 (63.03%)**	3592 (18.87%)	22.65

Table 3.1 A comparison of the number of spurious and missing minutiae detected by three algorithms: gradient-only [26], STFT-analysis [14], and the proposed method. The total number of minutiae as indicated by ground truth was 19032. The last column shows the equal error rates on the same dataset. We see that CRBMs are an improvement over Gabor-based enhancement.

3.6 Summary

We have demonstrated the use of continuous restricted Boltzmann machines in correcting orientation fields that have been estimated by a gradient method. We have moved a bit further in the direction of unsupervised feature learning applied to fingerprint orientation field estimation and correction. Possible directions for the future include converting to deep networks, and learning directly from greyscale images without an initial step of feature extraction. The database of gathered fingerprints makes this direction of research worth exploring, simply because of the sheer variety, number, types, and easy-availability of fingerprint images.

Chapter 4

Hierarchical Learning of Fingerprint Features

4.1 Introduction

Fingerprint recognition is the most widely used biometric to identify individuals. With a history of over a century, fingerprint recognition has been observed to be applied to a host of systems and tasks. The list includes authentication systems, forensic science, and academic institutions, to name a few. [47]

It is now apparent that fingerprint recognition tasks should be as robust as possible. As expected, much has already been achieved in this field ([47, 22, 14, 13, 60, 86, 5, 19, 44, 7]). Having started out as a manual task, where fingerprint experts would sit down with pairs of fingerprint images and try to find a match between them, recognition tasks have been taken over by computers since their advent. Advances in image processing have enabled development of robust algorithms, and technology has allowed these algorithms to be available to end-users for small costs. However, this easy availability has been possible only because extensive research has tackled challenges faced during fingerprint recognition. A prominent and ever-present challenge is tolerance of noise. The most widely available fingerprint sensors today induce at least one kind of noise in the images they output. This noise ranges from skin conditions such as dirt, injuries, cuts and bruises, and moisture, to that introduced by the sensors themselves, typically via. a worn-out sensor-surface [47]. It is essential, then, to remove these types of noises and extract the fingerprint image without losing information.

From the perspective of pattern recognition, this can be seen as presence of large intra-class variations - prints from the same finger can vary significantly, especially if taken via. different sensors. It is important, then, to clean up fingerprints before proceeding with the detection and matching of minutiae. Several approaches for this have been explored, and are referred to collectively as fingerprint enhancement.

4.2 Related Work: A Recall

The orientation field of a fingerprint image is very essential in enhancing the fingerprint. Filtering the image using Gabor filters based on ridge direction and frequency at a point gives a robust enhancement of the print. Since Hong, Wan and Jain proposed the use of Gabor filters, [26], several modifications of the approach have been developed. Zhu, Yin and Zhang [86] achieve faster enhancement using circular Gabor filters. Yang *et al.* [84] developed a modified version of this technique to obtain more consistent enhancement, without damaging fingerprint structure. Bernard *et al.* [5] use wavelet filtering to arrive at a multiscale approach. Algorithms which pursue other filtering techniques have been developed, but couldn't gather as much momentum as Gabor filters. Greenberg *et al.* [22], for instance, used Weiner and anisotropic filtering. However, the task of orientation field estimation still remains very important to fingerprint enhancement, especially for very noisy images and latent prints [19]. Several frontiers have been explored to achieve robust estimation of orientation fields. These can be broadly classified into location estimation and global models.

The most popular and significant algorithms falling in the former class is estimation using gradient-based methods [26]. These methods typically consider the neighbourhoods of a point while estimating the orientation field there. Hence, they are quite susceptible to noise in the image. A small distortion in ridge structure can cause an error in estimation of the orientation field, thus giving false enhancements.

To counter this, the orientation field generated by the gradient-based methods is subjected to a smoothing operation. Low-pass filtering is the most common form of smoothing, but its effectiveness is not up-to-the-mark when it comes to very noisy images.

Global models, on the other hand, try to model whole fingerprint structures, instead of using local ridge structures to estimate orientation fields. Popular approaches in this category employ the use of Markov random fields (MRFs) to arrive at a fingerprint structure that minimises energy. Some examples of these are Dass [16], and Lee and Prabhakar [13]. Rama Reddy and Namboodiri [60] improve upon these further by employing hierarchical MRFs which use loopy belief propagation.

Researchers have also explored the frequency domain, in contrast to the spatial domain, to enhance fingerprints. The most popular work in this category, by Chikkerur, Cartwright and Govindaraju [14], uses the analysis of short time Fourier transform. A given fingerprint image is divided into regions, and intrinsic properties of the fingerprint are estimated for these regions. Intrinsic properties include the orientation in those regions, the local ridge frequency, and the mask which indicates the presence of a fingerprint. Based on these, filters are constructed and instead of the convolution operation, an equivalent operation of matrix multiplication in the frequency domain is done, after which the final step is to take an inverse Fourier transform of the resulting image. Results of STFT analysis are comparable to Gabor-based enhancements, which opens up new prospects.

Although we have observed a lot of success with these approaches, we can conclude that all of them are 'supervised' in some sense. On the opposite side, few attempts have been made on unsupervised fingerprint enhancement or recognition. It can be argued, however, that supervised approaches have worked so well because they are specific to fingerprint image, and thus can be tailored specifically to

perform fingerprint recognition tasks. To move in the direction of unsupervised learning, we need robust models which perform competitively in feature learning. For instance, Leung, Engeler and Frank [43] incorporated the use of neural networks to extract fingerprint minutiae. The extraction of minutiae is performed on Gabor-filtered images. In another work, Pais Barreto Marques and Gay Thome [56] train MLPs on Fourier transformed patches. Some progress was also made by Altun and Allahverdi [2] in using neural networks for fingerprint recognition. In the previous chapter, we presented an approach which explored the correction of fingerprint orientation fields using restricted Boltzmann machines. We learnt patterns in orientation field images, and used CRBMs to reconstruct noisy regions. Results indicated an improvement over traditional Gabor-based algorithms in terms of both, matching and minutiae detection. However, such attempts have been far fewer compared to other methods.

In this chapter, we look to use a deep, generative neural network to achieve fingerprint image enhancement. Strides have been made in deep learning over the past decade, and a host of models suited to different tasks have emerged. Deep belief networks (DBNs) arose as suitable many-layered (deep) models to learn patterns. DBNs have the restricted Boltzmann machines (RBMs) as their building blocks. Each layer of the DBN is an RBM. As newer methods to train neural networks surfaced [8], it became more feasible to use neural networks in pattern recognition.

For this chapter, we were interested in a generative, deep model that is capable of working directly on pixels and extracting features in an unsupervised manner. Convolutional networks fit the choice very well, in that they are scalable to large images, and feature detection performed by them is robust ([41]) and they now are known to perform very well at several real-world classification problems ([73, 21]). However, convolutional neural networks themselves aren't generative models. Instead, we use the convolutional deep belief network (CDBN) for our task. Desjardins and Bengio [17], developed convolutional RBMs, and were closely followed by Lee *et al.* [42], who introduced probabilistic max pooling and hierarchical probabilistic inference - both of which prove very essential in making the convolutional DBNs generative models. A convolutional RBM has several feature detectors working on the visible layer, which generate feature maps. The feature detectors (weights) work on entire images, in that the same weight is convolved with the entire image which introduces translational equivariance. Further, a pooling layer sits on top of the feature maps. Several such convolutional RBMs stacked on top of each other make up a convolutional DBN [42]. Unless otherwise stated, we will use the acronym CRBM to mean *Convolutional Restricted Boltzmann Machine*, and CDBN to mean *Convolutional Deep Belief Network*.

4.3 Convolutional Deep Belief Networks

To maintain continuity in this chapter, we give a brief description of the convolutional deep belief networks in this section. We will start with the basic building block for a convolutional DBN - the convolutional RBM - and move on to the DBN, later to computing the network's representation of the image, and then reconstructing from this representation using the learnt parameters.

4.3.1 Convolutional RBM

A convolutional RBM is a neural network that, like an RBM, has a visible layer and a hidden layer. In addition to an RBM, though, it also has a pooling layer. The visible layer of an RBM is shown images from which features need to be learnt. This data, called *training data*, is a very essential factor in learning representations. A detailed description of the training data we used for our learning is given in Section 4.4.1.

From images shown to the visible layer, the network draws an inference, in that it computes states of neurons in the hidden layer. Let the visible layer be called V, and have a size of $N_V \times N_V$ (Figure 4.1). The hidden layer, H, consists of K groups of units, each N_H pixels $\times$ N_H pixels in size. With each group of hidden units, $\mathbf{h}^k$, we associate a group of pooling units, $\mathbf{p}^k$. The size of this group of units is $1/q^2$ times the size of the group of hidden units (q usually being 2 or 3), and each pooling unit is connected to a $q \times q$ contiguous block of units in the hidden group. Two pooling units never have overlapping blocks, and the union of all blocks is all of the hidden units. Further, with each group of hidden units, we associate a weight that connects the visible units with the group. Call this weight W^k, where k refers to one of the K groups of units. If $\mathbf{v}$ denotes the values of visible units, then the energy of the network is given by

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} \mathbf{h}^k \bullet \left(\tilde{W}^k * \mathbf{v}\right) - \sum_{k=1}^{K} \left(b_k \sum_{i,j} h_{i,j}^k\right) - c \sum_{i,j} v_{i,j} \tag{4.1}$$

where b_k is the bias associated with the k-th group of hidden units, whereas all the visible units share a single bias c. Here, $*$ is used to mean convolution, $\tilde{W}$ represents a horizontally- and vertically-flipped W, and $\bullet$ denotes element-wise product of two matrices followed by summation of elements of the result.

Neurons in the k-th group of hidden units are obtained by a 'valid' convolution of the visible layer with the weight W^k. Hence, the size of the weight is related to the sizes of the visible and hidden layers according to the formula $N_W = N_V - N_H + 1$. Similar to the RBM, values of visible neurons are also computed from the hidden neurons, but this step involves a full convolution. Now, to get activations of neurons in the hidden layer, we define a probability distribution which determines probabilities for each neuron's ON state. To define this probability, we first look at the input a neuron in the k-th group of hidden units receives from the visible layer:

$$I\{h_{i,j}^k\} = \left[\tilde{W}^k * v\right]_{i,j} + b_k \tag{4.2}$$

Here, $[X]_{i,j}$ is used to refer to the (i,j)-th pixel in the image X. The neurons in this group of hidden units are then separated into disjoint blocks of size $q \times q$. Each block in this group of units is connected to exactly one unit in the k-th group of pooling units. Call this block of hidden units α, and the said pooling unit p_α^k. Let $B_\alpha^k = \{(x,y) | h_{x,y}^k \in \alpha\}$. The probability that a neuron (i,j) in this block is ON is given by:

$$P(h_{i,j}^k = 1 | \mathbf{v}) = \frac{\exp(I\{h_{i,j}^k\})}{1 + \sum_{i',j' \in B_\alpha^k} \exp(I\{h_{i',j'}^k\})} \tag{4.3}$$

Further, at most one unit from the block B_α^k can fire at a time. To achieve this, sampling is done from a multinomial distribution given by the probabilities $P(h_{i',j'}^k)$, $(i', j') \in B_\alpha^k$. This gives $q^2 + 1$ possible states of the block B_α^k - one each for one of the q^2 units being ON, and one state where none of the units fire. The latter has a probability $1 - \sum_{i',j' \in B_\alpha^k} P(h_{i',j'}^k)$. The pooling unit p_α^k takes a value of either 1 or 0, with the value being 1 if any of the units in B_α^k are ON. The probability that this pooling unit fires is, hence,

$$P(p_\alpha^k = 1|\mathbf{v}) = \frac{\sum_{i',j'} \exp(I\{h_{i',j'}^k\})}{1 + \sum_{i',j' \in B_\alpha^k} \exp(I\{h_{i',j'}^k\})} \tag{4.4}$$

This method of sub-sampling the hidden units to get pooled units is called *probabilistic max-pooling*. Unlike other pooling algorithms used mostly in convolutional neural networks (for example, *max-pooling*), this method of sub-sampling is probabilistic in nature. That is, given the same scenario of hidden units, different samplings might generate different hidden units. It was observed ([42]) that this strategy works better than max-pooling in making the convolutional deep belief network a generative model.

We perform alternate Gibbs sampling to get the activations of visible and hidden neurons. Training is done with an update to the weights that is proportional to the difference of positive and negative associations, like the RBM. The associations corresponding to a weight are calculated by convolving the visible layer, which is the same for all weights, with the group of hidden units corresponding to that weight. To approximate the objective function, contrastive divergence learning with 1 step (CD-1) was employed throughout this chapter, unless otherwise stated. Hence, we compute $\mathbf{h}^{(0)}$ from $\mathbf{v}^{(0)}$, and also compute $\mathbf{h}^{(1)}$ and $\mathbf{v}^{(1)}$ in the next step of alternate Gibbs sampling. The update to weight W^k is then given by

$$\Delta W^k \propto \frac{1}{N_H^2} \left(\mathbf{v}^{(0)} * \mathbf{h}^{(0),k} - \mathbf{v}^{(1)} * \mathbf{h}^{(1),k} \right) \tag{4.5}$$

where $\mathbf{h}^{(n),k}$ denotes the k-th set of hidden units at the n-th step of alternate Gibbs sampling. Likewise for $\mathbf{v}^{(n)}$.

To ensure that the representation that we learn is sparse, a sparsity update is done at every step of learning. This is defined as

$$\Delta b_k^{sparsity} \propto s - \frac{1}{N_H^2} \sum_{i,j} P(h_{i,j}^k|\mathbf{v}), \tag{4.6}$$

where s is the target sparsity. This update is added to the update to the hidden biases:

$$\Delta b_k \propto \frac{1}{N_H^2} \sum_{i,j} \left(h_{i,j}^{(0),k} - h_{i,j}^{(1),k} \right) + \Delta b_k^{sparsity} \tag{4.7}$$

Recall that hidden and pooling units are calculated using probabilistic max pooling. This method defines a multinomial distribution for every block, which dictates probabilities of being ON for each of the units in that block. Also, this probability uses the $\exp$ function on the total input received at a unit. Since we add the hidden bias to the total input received, and $\exp$ is a monotonically increasing function, using smaller values of the bias will lead to lower probabilities. This ensures that few units are ON at a time,

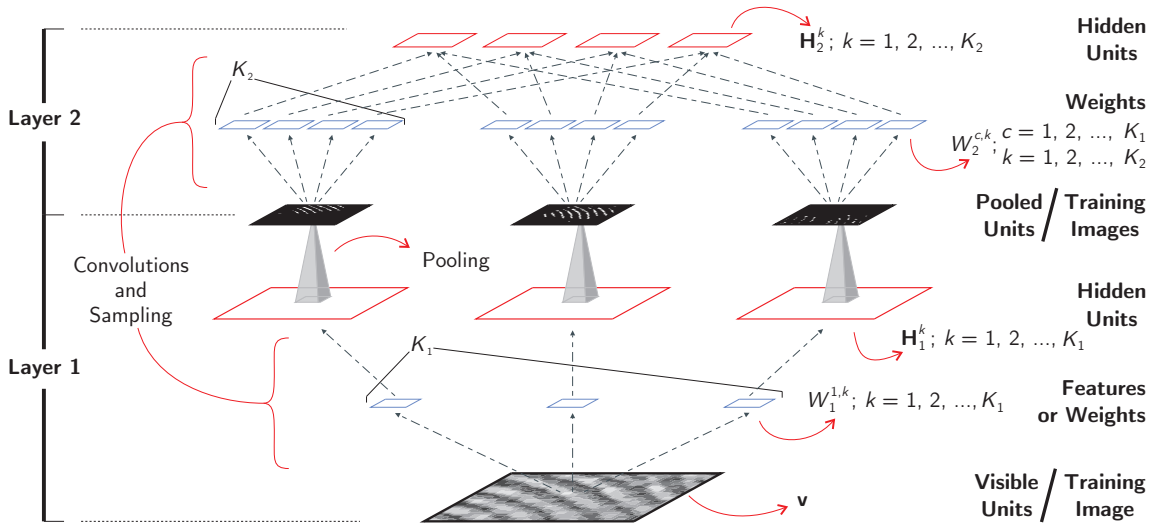

Figure 4.1 A convolutional deep belief network, with 2 layers. The further layers work on the data that is found in the previous layer's pooling units. The notation $W_c^{a,b}$ refers to the b-th weight corresponding to the a-th channel in the c-th layer, where a *channel* is one of the components of the input. We use greyscale images and so have only one channel in the visible units of the first layer

and as the hidden units directly reflect in the weight updates, we find that a sparsity update of the given type to the hidden biases learns a sparse representation.

4.3.2 Converting to a Deep Network

With this building block at our disposal, we can now stack multiple convolutional RBMs on top of each other to build a convolutional DBN. We refer to each such stacked convolutional RBM (CRBM) as a *layer* in the convolutional DBN (CDBN). In such a model, the pooling units of a layer in the DBN (which is a CRBM) serve as visible units for the next layer. As stated earlier, a value of 2 or 3 for q is used. This sub-samples the input images by a factor of q, and enables higher layers of the convolutional DBN to learn higher-level features. Quite predictably, we observe the same results when the convolutional DBN is trained on a set of fingerprint images.

Training the convolutional DBN involves layer-wise training of each layer using the algorithm stated in the previous section. Once this training is complete, pooling units are generated using the learnt weights on complete images.

4.3.3 Hierarchical Probabilistic Inference

Hierarchical probabilistic inference (HPI) is an algorithm to reconstruct an image using the network's representation of it. The term *hierarchical* says we use data from higher layers to affect values of hidden units in lower layers, and in-turn the reconstructions of images. The algorithm is *probabilistic* because it uses probabilistic max-pooling, described in Section 4.3.1, to draw an inference. To use data from

higher layers, we modify hidden neurons in a layer to receive inputs from the layer's visible units, as well as from the next layer's hidden units. This redefines the probability of a neuron firing.

Consider a layer, l, in the network. To get the top-down signal from layer $l+1$, we calculate a full convolution between the next layer's hidden units and its weights. This gives us an image the size of l's pooling units. Now, each neuron in this group of units forwards the signal it gets to the $q \times q$ sized group of hidden units of layer l that it connects with. Call $\mathbf{H}_l^k$, the k-th groups of layer l's hidden units. Let $W_l^{c,k}$ represent the weight corresponding to the c-th channel in the k-th set of hidden units of the l-th layer. Then the top-down signal received by $\mathbf{H}_l^k$ is (to improve readability, we denote $\mathbf{H}_l^k$ by $\mathbf{h}'$ in the next three equations. The (i,j)-th unit in $\mathbf{H}_l^k$ can then be denoted by $\mathbf{h}'_{i,j}$):

$$T\{\mathbf{h}'_{i,j}\} = \left[\sum_{r=1}^{K_{l+1}} \mathbf{H}_{l+1}^r * W_{l+1}^{k,r}\right]_\alpha \tag{4.8}$$

where the (i,j)-th unit is in block B_α of $\mathbf{h}'$. We now get new probability distributions for $\mathbf{h}'_{i,j}$ and p_α^k by substituting $I\{\mathbf{h}'_{i,j}\}$ with $I\{\mathbf{h}'_{i,j}\} + T\{\mathbf{h}'_{i,j}\}$ in Equations 4.3 and 4.4:

$$P(\mathbf{h}'_{i,j} = 1 | \mathbf{v}_l, \mathbf{H}_{l+1}) = \frac{\exp(I\{\mathbf{h}'_{i,j}\} + T\{\mathbf{h}'_{i,j}\})}{1 + \sum_{i',j' \in B_\alpha^k} \exp(I\{\mathbf{h}'_{i,j}\} + T\{\mathbf{h}'_{i,j}\})} \tag{4.9}$$

and

$$P(p_\alpha^k = 1 | \mathbf{v}_l, \mathbf{H}_{l+1}) = \frac{\sum_{i',j'} \exp(I\{\mathbf{h}'_{i,j}\} + T\{\mathbf{h}'_{i,j}\})}{1 + \sum_{i',j' \in B_\alpha^k} \exp(I\{\mathbf{h}'_{i,j}\} + T\{\mathbf{h}'_{i,j}\})} \tag{4.10}$$

where p_α^k is the pooling unit corresponding to the (i,j)-th unit in $\mathbf{h}'$.

These equations establish a dependence between the hidden units of a layer on one side, and the layer's visible units and the next layer's hidden units on the other. Hence, hierarchical probabilistic inference is computed using block Gibbs sampling, in which we compute all quantities dependent on one set of quantities in one half of an iteration, and vice versa in the other half. Appendix B describes this process in more detail on a two-layered and a three-layered network.

4.4 Enhancing Fingerprint Images Using a CDBN

In this section, we will describe the training and use of a CDBN for fingerprint image enhancement.

4.4.1 The Training Data

The training data used is very important to learning. The network learns features depending on the training data. The reconstruction performed by this network also relies heavily on the training data. As we're trying to perform reconstruction of fingerprint images which should serve as enhancement, we use noise-free fingerprint images to train our network. We hand-picked images from standard fingerprint datasets (FVC 2000 Db1_a [45], FVC 2002 Db1_a [46], and NIST-4 special database [77]).

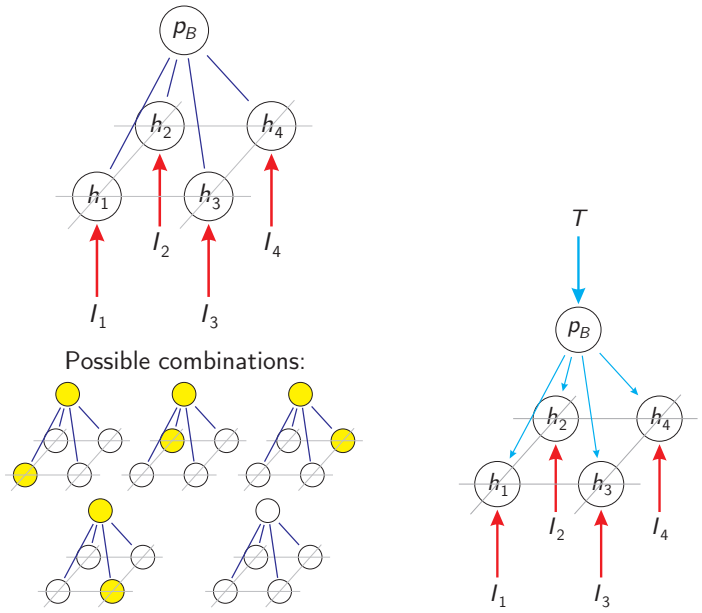

Figure 4.2 *left:* Bottom-up and; *right:* combination of bottom-up and top-down inferences

While selecting images, we followed some rules: (a) significant variety in ridge orientations and ridge frequencies should be present in the image; (b) the images should have the least patches of noise, i.e., inconsistent ridges, cuts, bruises, smudges, wet fingers, pores, dirt, stray ink, lettering, and noise from sensors; (c) the images should have uniform contrast; and (d) the images should have uniform moisture, and shouldn't be too wet or too dry. Further, the selected images were trimmed so as to include minimal regions that don't belong to the fingerprint. Before being used for training, the training data was normalised [47, 26] and whitened [55] before it is fed to the network for training. Whitening ensures that the training images have equal variance in different directions, and hence helps the gradient descent learning in its search for a solution.

Our training data consisted of 15 different images of varying sizes, and resolutions. Due to repetitive patterns in fingerprint images, a small number of images is enough to learn a model that can reconstruct test images.

4.4.2 Training the CDBN on Fingerprint Images

We train the convolutional deep belief network on the preprocessed images. We used a two-layered network with greedy layer-wise training. Greedy layer-wise training refers to training each layer of the convolutional DBN by considering it to be a convolutional RBM. We start with the first layer, and the training data for this layer is our set of training images. When the first layer is trained, training data for the next layer is generated. This is done by calculating the first layer's representation of every greyscale training image, and pooling these units to get the training data for the second layer. The same process is then done with the second layer. To reconstruct the entire image, we used hierarchical probabilistic

inference. The layer-wise training isn't done on complete images, but on patches randomly chosen from these images. This choice considerably quickens the training process and also leads to better features being learnt. It is easy to see that this choice doesn't affect the learnt filters either. To emphasise, the sparsity update to the bias, $b_k^{sparsity}$, is calculated so that it is averaged over N_H^2 hidden units. As the number of hidden units is directly defined by the size of the input images and the sizes of the filters we are learning in a layer, and we have averaged the update to the bias over the hidden units, the size of the input images doesn't affect the learning. We only have to ensure that the size of the input images for a layer is adjusted so that we have a considerable amount of fingerprint structure in the area that the filters in that layer operate on. Hence, the size of the input images should always be greater than the size of the learnt filters. The initial weights are drawn from normally distributed random numbers and then multiplied by a factor of 0.01. Further, the visible biases are set to zero initially, and the hidden biases are also drawn from normally distributed random numbers but are multiplied with a factor of -0.1.

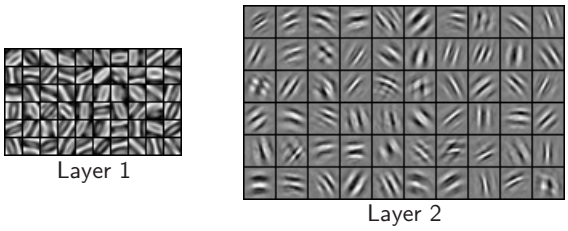

Figure 4.3 Weights learnt by the first and second layers. The training data contained fingerprints with various ridge frequencies, which is visible in the diversity in ridge frequencies in the learnt weights.

The first layer learns single, oriented ridges. The weights learnt in this layer are similar to Gabor and other oriented filters that are used for fingerprint image enhancement. We empirically chose the number of feature maps in the first layer to be 60 because the training has combinations of many orientations and frequencies. The weights learnt by the first layer are shown in Figure 4.3. The size of weights used in the first layer was 10 pixels $\times$ 10 pixels. To achieve this set of weights, we used a target sparsity of 0.003.

The second layer learnt higher level features than the first. This is because of a pooling operation on the hidden units of the first layer which down-samples the image at the hidden units. This factor by which the image is down-sampled was set to 2 in our implementation. Further, the second layer had 60 features, each of size 10 pixels $\times$ 10 pixels. A weight now covers a larger portion of the input image. Weights learnt in the second layer are visualised in Figure 4.3. A target sparsity of 0.005 was used for this layer. The second layer is essential in drawing hierarchical probabilistic inference of an image, and "filling up" regions of the fingerprint image which couldn't be reconstructed by the first layer alone.

4.4.3 Visualising the Features in Higher Layers

Let K_l be the number of designated weights in layer l. A layer l ($l > 1$) will actually learn more than K_l weights. As the input to this layer is multi-channelled, the specified number of weights are learnt

for each channel (we didn't notice this in the first layer because our input consisted only of greyscale images). Hence, the l-th layer trains $K_{l-1} \cdot K_l$ kernels. However, the learning that happened in the l-th layer is seldom understood by viewing each of these $K_{l-1} \cdot K_l$ matrices. Instead we represent these kernels in a better manner - as a combination of weights from lower layers. The second layer features are, hence, visualised as a combination of the weights in the first layer. The m-th feature in the visualisations is given by $\Omega_2^m = \sum_{k=1}^{K_1} W_2^{k,m} * W_1^{1,k}$; $1 \leq m \leq K_2$, where $W_l^{c,k}$ represents the weight corresponding to the c-th channel in the k-th basis of the l-th layer; and $*$ represents a full convolution (we have only one channel in the first layer).

4.4.4 Reconstructing Fingerprint Images Using a Learnt Model

Now that we have trained a model, we can use it to reconstruct fingerprint images. These reconstructions serve as the enhancement which we are targeting. To reconstruct an image, we show it to the visible units of the first layer, and calculate the network's representation of it. We then reconstruct the image from this representation using hierarchical probabilistic inference, in which we iterate over 20 steps of block Gibbs sampling. More information on the actual reconstruction steps can be found in Appendix B.

Once we have completed 20 iterations of block Gibbs sampling in hierarchical probabilistic inference, we look at the hidden units of the first layer. The enhanced image is generated using only these units and the weights of the first layer using the following equation:

$$\mathbf{v}' = \sum_{k=1}^{K_1} W_1^{1,k} * \mathbf{H}_1^k \tag{4.11}$$

where $\mathbf{H}_1^k$ are the activations of the k-th set of hidden neurons in the first layer after 20 iterations of block Gibbs sampling.

We observe the reconstructions computed by the network at every iteration of block Gibbs sampling, and we see that the second layer progressively reconstructs regions that the first layer alone was unable to represent (Figure 4.4).

4.4.5 Estimating Orientation Field, Frequency Image and Region Mask

Now that we are able to perform a reconstruction of the fingerprint image using the convolutional deep belief network, we show a method to estimate intrinsic images of the fingerprint, viz., the orientation field the frequency image, and the region mask. Just like the reconstruction of the image itself, we show that hierarchical probabilistic inference can help in reconstructing the orientation fields in regions where the the first layer alone couldn't perform a reasonable reconstruction. To estimate these images, we look at weights learnt in the first layer. These are very close to oriented filters used for enhancement, and hence have a ridge orientation and ridge frequency associated with them. The associated orientation and frequency for a weight can be found using the Fourier transform of the weight. To get a reasonable

association of a weight with an orientation and frequency, we calculate a weighted average of the orientations and frequencies given by the Fourier transform, weighed according to the energy of the Fourier transform corresponding every orientation. Chikkerur, Cartwright, and Govindaraju [14] show a method to use the Fourier transform for this task. We borrowed from their method to associate orientations and frequency with a weight. If Φ_k is the shifted Fourier transform of weight $W_1^{1,k}$ and θ is the matrix signifying the orientation angle for every point in Φ_k, the orientation associated with $W_1^{1,k}$ is given by

$$d_k = \frac{1}{2} \arctan \frac{\sum \sin\left(2\theta \cdot |\Phi_k|^2\right)}{\sum \cos\left(2\theta \cdot |\Phi_k|^2\right)}, \tag{4.12}$$

where $\cdot$ is element-wise multiplication of two matrices.

Frequency, f_k, for a filter can be calculated from the Fourier transform too. We use the distance of peaks in the power spectrum from the centre - R - instead of θ.

$$f_k = \frac{1}{W_s} \frac{\sum R \cdot |\Phi_k|^2}{\sum |\Phi_k|^2}, \tag{4.13}$$

where W_s is the size of a weight in the first layer, and also the size of the computed Fourier transform of a weight. Once we have associated an orientation and a frequency with every weight in the first layer, we perform hierarchical probabilistic inference. This involves 20 steps of block Gibbs sampling, after which we have activations and states of hidden units of the first layer. The reconstruction, $\mathbf{v}'$, of an image is now given by

$$\mathbf{v}' = \sum_{k=1}^{K} W_1^{1,k} * \mathbf{H}_1^k \tag{4.14}$$

To calculate the orientation field, we calculate a weighted average of the orientations from K groups of hidden units, weighed according to $W_1^{1,k} * \mathbf{H}_1^k$. However, instead of averaging d^k, we average $\sin(2d^k)$ and $\cos(2d^k)$. This ensures that we preserve the allowed range of the orientation, which is $[0, \pi[$, and also maintains continuity in the orientation field, as it jumps abruptly from 0 to π at some places (the apices of downward-curved ridges, for instance). We then retrieve the orientation field using an inverse tangent operation on the results of these two averages. An estimate of the orientation field is obtained using Equations 4.15, 4.16, and 4.17:

$$\mathbf{D}_s = \left(\sum_{k=1}^{K} \sin(2d_k) \left(W_1^{1,k} * \mathbf{H}_1^k\right)\right) \oslash \left(\sum_{k=1}^{K} W_1^{1,k} * \mathbf{H}_1^k\right); \tag{4.15}$$

$$\mathbf{D}_c = \left(\sum_{k=1}^{K} \cos(2d_k) \left(W_1^{1,k} * \mathbf{H}_1^k\right)\right) \oslash \left(\sum_{k=1}^{K} W_1^{1,k} * \mathbf{H}_1^k\right); \tag{4.16}$$

$$\mathbf{D}' = \frac{1}{2} \arctan\left(\mathbf{D}_s \oslash \mathbf{D}_c\right), \tag{4.17}$$

where $\oslash$ is element-wise division. Finally, we smoothen the complete orientation image using a 5×5 Gaussian smoothing kernel. This is done by smoothing sine and cosine images obtained from the

orientation field $\mathbf{D}'$, and then combining the two using an inverse tangent operation. The final orientation field image, $\mathbf{D}$, is given by:

$$\mathbf{D} = \frac{1}{2}\arctan\left(\frac{G * \sin(2\mathbf{D}')}{G * \cos(2\mathbf{D}')}\right) \tag{4.18}$$

The frequency image can be estimated in a similar way, using f_k instead of d_k:

$$\mathbf{F}' = \left(\sum_{k=1}^{K} f_k \left(W_1^{1,k} * \mathbf{H}_1^k\right)\right) \oslash \left(\sum_{k=1}^{K} W_1^{1,k} * \mathbf{H}_1^k\right) \tag{4.19}$$

We smoothen the estimated frequency image $\mathbf{F}'$, using isotropic diffusion, as proposed by [14]:

$$\mathbf{F}_\alpha = (\mathbf{F}'_\alpha \cdot G) \oslash G;\ \forall\alpha \tag{4.20}$$

where α represents corresponding blocks in $\mathbf{F}'$ and $\mathbf{F}$, and G is a Gaussian smoothing kernel. This gives a smoothened frequency image.

The region mask can be estimated in a similar manner. Note that the network performs reconstructions of only those regions that match a learnt fingerprint pattern. Hence, the region mask can be thought of consisting of all those pixels where the convolutional DBN performs reconstruction. Hence, a pixel (x, y) is set to 1 in the region mask, $\mathbf{M}$, if there is a response from the network at that pixel. However, considering that there might be other points in the image where pixel values will be zero (for example, pixels where reconstruction was done, but the convolution resulted in a pixel-value of zero or near-zero), we perform a morphological operation on the detected mask to remove such stray pixels. Instances of such pixels are often "sprinkled" on the region mask obtained using Equation 4.21, much like salt-and-pepper noise is scattered over an image. Hence, closing the mask using a 2×2 matrix of ones helps in removing these stray pixels. The result of this operation is a uniform region mask which determines where in the image the fingerprint is.

$$[\mathbf{M}]_{x,y} = \begin{cases} 1; & \text{if } [\mathbf{v}']_{x,y} \neq 0 \\ 0; & \text{if } [\mathbf{v}']_{x,y} = 0 \end{cases} \text{; where } \mathbf{v}' = \sum_{k=1}^{K} W_1^{1,k} * \mathbf{H}_1^k \tag{4.21}$$

4.5 Experimental Results

Performance of an enhancement algorithm in terms of fingerprint matching is very essential to its importance. How an enhancement algorithm fares on fingerprint matching is crucial to its acceptability. This, of course, means that the operation of enhancement on a fingerprint should not destroy the individuality and distinctiveness of the fingerprint. The ability of an enhancement algorithm to maintain them is judged using how the enhanced images score on genuine and impostor comparisons. Given a set of several impressions of the same finger, a comparison performed between two of these impressions using a matching algorithm is called a genuine comparison. Matching with impressions that aren't of the same finger is also important, and such comparisons are called impostor comparisons. In other words, these comparisons count the number of false positives and false negatives. Ideally, an enhancement algorithm

should enable the fingerprint matcher to provide high scores on genuine comparisons, and low scores on impostor comparisons. In this section, we evaluate our model in terms of the performance of a matching algorithm on it. We will also compare our results with existing Gabor-based and STFT-based enhancement algorithms. We will plot Receiver Operating Characteristics (ROC) for the performances on some standard FVC (Fingerprint Verification Competition) datasets ([45, 46]).

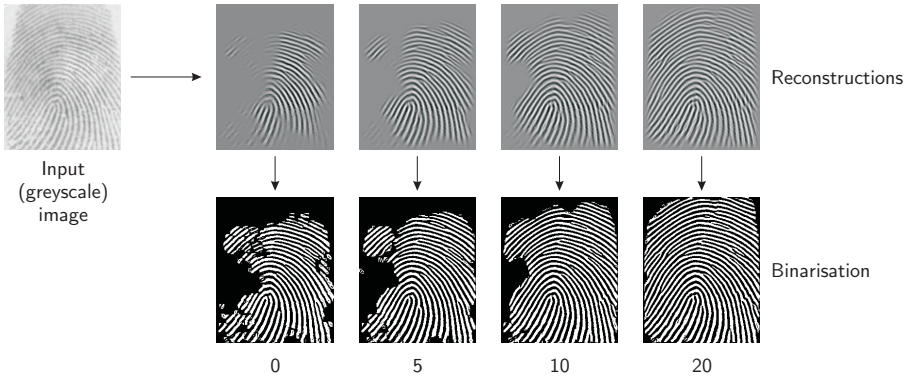

Figure 4.4 The network's reconstruction of a fingerprint image as the number of iterations of alternate Gibbs sampling in hierarchical probabilistic inference increases. The representation of the input in the hidden units of a layer is affected by both - the bottom-up inference coming from the layer's visible units, and the top-down inference coming from the next layer's hidden units. The figure shows reconstructions at various iterations. 0 iterations correspond to reconstruction using only the first layer. The images in the first row are reconstructed images, and they were thresholded based on their mean values to give the images in the second row.

4.5.1 Experimental Setup

The convolutional deep belief networks used in this chapter were programmed in MATLAB. The program was run on a Linux system with a 3.3 GHz quad-core processor and 4 GB of memory. The layers were trained on patches of size 50 pixels $\times$ 50 pixels randomly chosen from 15 training images, which themselves were of different sizes. Training was done for 500 epochs at each layer. We show in this section that the model proposed in this chapter fares well on these tests.

4.5.2 Qualitative Analysis

We perform a qualitative analysis of our algorithm on images from three datasets: FVC2000 Db1 and Db2 [45], and FVC2002 Db3_a [46]. Images from the dataset were shown to the trained convolutional deep belief network, and reconstructions of these images were recorded.

Figure 4.4 is an example of the reconstruction performed by a trained network. "CDBN-k" denotes the reconstructions obtained using k iterations of block Gibbs sampling in HPI. The binary images in the figure have been generated by thresholding the reconstructed images according to their mean values. Pixel values greater than the mean constitute valleys, while the rest of them constitute ridges.

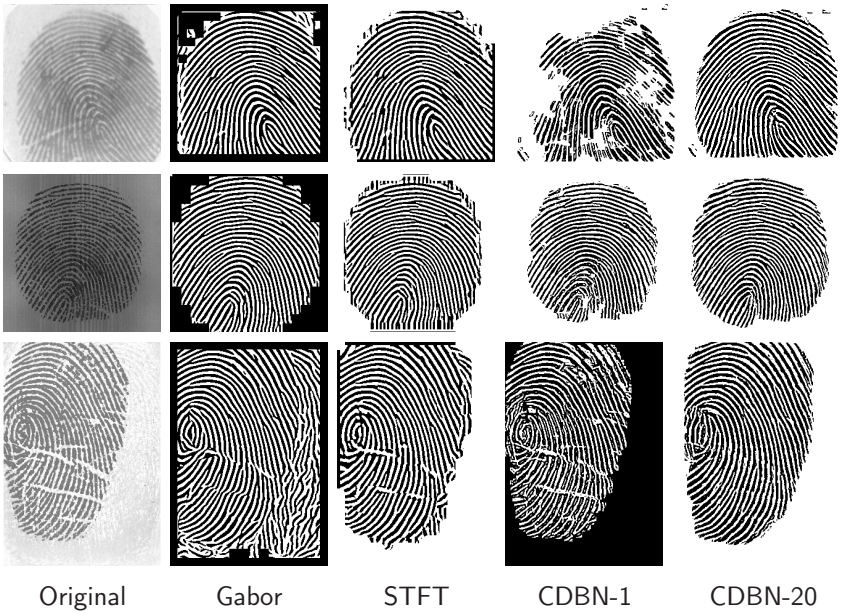

Figure 4.5 A comparison of enhancements and binarisations performed using Gabor [26], STFT analysis [14], CDBN-1 and CDBN-20 ("CDBN-k" signifies reconstructions obtained after k iterations of Gibbs sampling in hierarchical probabilistic inference). Each row shows performance on the left-most image.

Figure 4.5 shows a comparison between Gabor-based enhancement, STFT analysis, and reconstructions using the CDBNs for 1 and 20 iterations of Gibbs sampling in hierarchical probabilistic inference. We see that using higher-level data, we can "fill-up" regions of the fingerprint which the first layer is not able to comprehend. The first layer is "helped" by the higher layers in filling up this data. However, it is important that this filling up shouldn't destroy information of minutiae present in the input image. To achieve this, we propose a modification to the hierarchical probabilistic inference (HPI) algorithm. HPI involves partitioning all variables in all the layers into two disjoint sets and performing block Gibbs sampling on these variables (Section 4.3.3). We propose that during each iteration of block Gibbs sampling, the visible units be replaced by the input image, instead of using the values obtained from other units in the network. The reason for this is discussed in more detail in Appendix B. We saw that this modification to the reconstruction algorithm keeps the fingerprint structure intact, and tremendously improves enhancement. We also don't get spurious minutiae, which result from incorrect enhancements of fingerprints.

Using HPI, we are able to remove minor creases and ridge discontinuities from fingerprint images. The hierarchical inference also smoothens ridges in the enhanced images by removing a majority of the pores.

4.5.3 Quantitative Analysis: Evaluation Using Fingerprint Matching

The proposed algorithm was evaluated by conducting a matching exercise on fingerprints from standard datasets enhanced using the network. We used two datasets: FVC2000 Db1_b and Db2_b [45]. For minutiae extraction, we used the software FingerJetFXOSE developed by Digital Persona Inc. [1]. Fingerprint matching was performed using the NIST matcher, *bozorth3* [50]. To compare our approach with existing techniques, we have also stated the performance of Gabor-based enhancement [26] and short time Fourier transform analysis [14] on the same datasets. The choice of these two algorithms was motivated by their nature. Gabor-based enhancement and STFT-analysis are standard enhancement algorithms. Further algorithms usually start with either of these as building blocks. We propose that the algorithm stated in this thesis be counted as a different point of attack to the enhancement problem, and hence the comparison. Receiver operating characteristics for the three algorithms are given in Figure 4.6. This graph describes reconstructions using the CDBN for four values of number of iterations in HPI.

Equal error rates for the three approaches on these two datasets are tabulated in Table 4.1. A lower equal error rate indicates better accuracy of the corresponding enhancement algorithm. The equal error rate is the rate at which percent of false accepts are equal to the percent of false rejects, and have become the accepted measurement criterion for biometric systems. We observe that the reconstructions performed by the CDBNs are significantly better than Gabor enhancement and STFT analysis on FVC 2000 Db1_a, and comparable with these two algorithms on FVC 2000 Db2_a. We also observe that introducing more layers in the convolutional DBN and more iterations in hierarchical probabilistic inference gives substantially better results (CDBN-1 refers to reconstructions using only the first layer, while CDBN-20 refers to reconstructions using the second layer and twenty iterations of hierarchical probabilistic inference).

Dataset\Algorithm	**Gabor**	**STFT**	**CDBN-20**	**CDBN-10**	**CDBN-5**	**CDBN-1**
2000 Db1_a	8.47	8.79	6.62	8.19	9.59	10.57
2000 Db1_b	9.46	6.67	5.82	6.55	11.65	13.11
2000 Db2_a	7.71	7.98	8.52	9.14	10.24	10.66
2000 Db2_b	14.00	9.24	10.44	17.32	16.29	19.65
2002 Db3_a	24.34	21.99	23.95	25.00	25.45	24.48

Table 4.1 Equal error rates (in percentage) for the performance of the algorithms mentioned above on FVC 2000 Db1_a, Db1_b, Db2_a, Db2_b[45], and FVC 2002 Db3_a[46]. Lower equal error rates indicate better performance.

4.5.4 Quantitative Analysis: Evaluation Using Minutiae Count

Besides evaluating our approach using a fingerprint matching exercise, we conducted an evaluation using the detected minutiae too. The enhanced images were subject to feature extraction using Fin-

[1] `https://github.com/FingerJetFXOSE/FingerJetFXOSE`

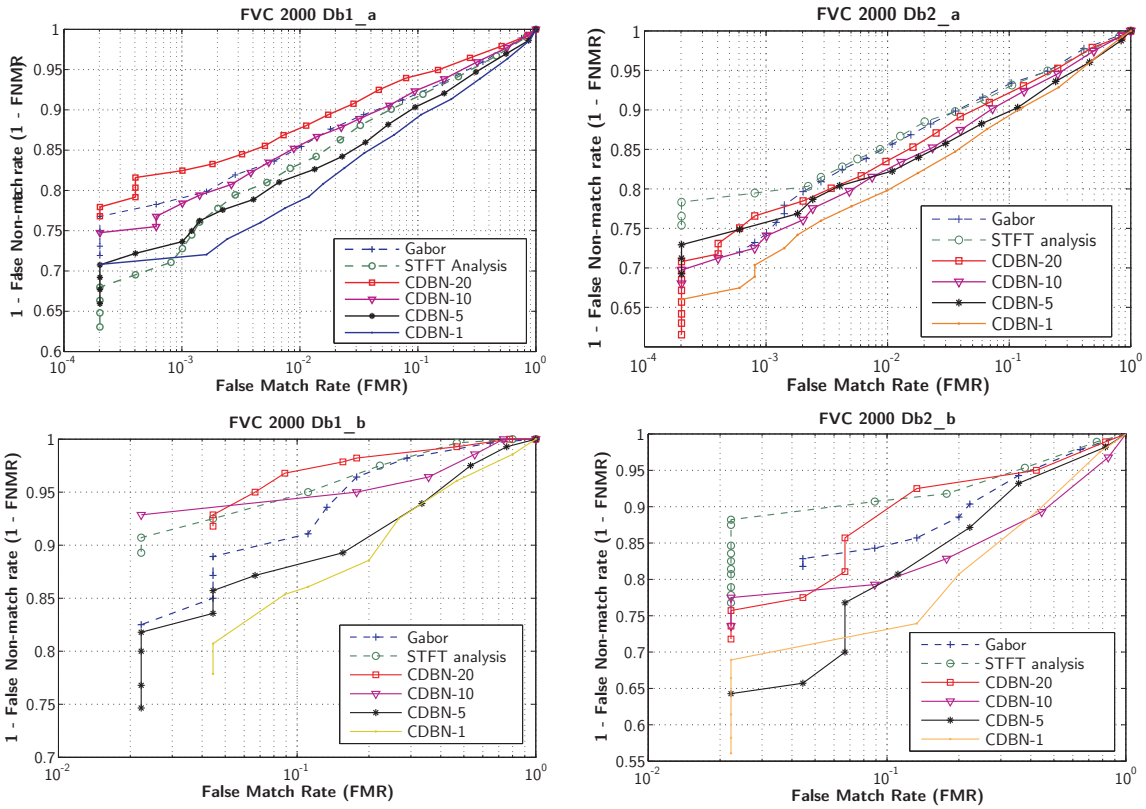

Figure 4.6 Receiver Operating Characteristics (ROC) plotted for the proposed algorithm, and compared with Gabor [26] and STFT-analysis [14]. This graph corresponds to performance on the (clock-wise from top-left) FVC 2000 Db1_a, FVC 2000 Db2_a, FVC 2000 Db1_b, and FVC 2000 Db2_b datasets. Plots for reconstructions from different number of iterations in hierarchical probabilistic inference are also shown for comparison ("CDBN-k" represents ROC curves from reconstructions obtained using k iterations).

gerJetFXOSE (referenced in the previous section), and a count of the total detected minutiae, and total spurious and missing minutiae extracted was made. We performed this exercise on the FVC 2002 Db3_a [46] dataset. To count the number of spurious and missing minutiae, we used ground truth data provided by Umut Uludag [36] for this dataset. We record the observed values in Table 4.2.

It is interesting to note that images from FVC 2002 Db3_a were not a part of the training set used to train the CDBN. Further, FVC 2002 Db3_a is a dataset with relatively noisy fingerprints. Our approach using the convolutional deep belief network detected fewer minutiae than either STFT analysis or Gabor-based enhancement. It also performed at-par with Gabor enhancement and STFT analysis in terms of performance on a matching exercise. The equal error rates for the proposed approach on FVC 2002 Db3_a are tabulated in Table 4.1.

Algorithm	Ground Truth	Detected	Spurious	Missing
Gabor	**19032**	53389	38415 (71.95%)	4058 (21.32%)
STFT-analysis		48963	33096 (67.59%)	**3165 (16.63%)**
CDBN-20		**31151**	**16211 (52.04%)**	4092 (21.50%)

Table 4.2 A comparison of the number of spurious and missing minutiae detected by Gabor-based enhancement [26], STFT analysis [14], and the proposed approach on the FVC 2002 Db3_a dataset.

4.5.5 Varying the Number of Features in Layers

The number of features, or weights, in every layer plays a crucial role in the reconstructions achieved by that layer. A higher number of weights directly means that more number of features will be learnt from the training data, and hence, we will achieve better reconstructions. However, it is also essential to show that this holds true based on quantitative analysis. We experimented with several networks, varying the number of weights in each one of them. We compare our results with two more networks - one with 20 weights in layer 1 and 30 weights in layer 2, and another with 40 weights in layer 1 and 60 weights in layer 2. The weights learnt by these networks are displayed in Figure 4.7.

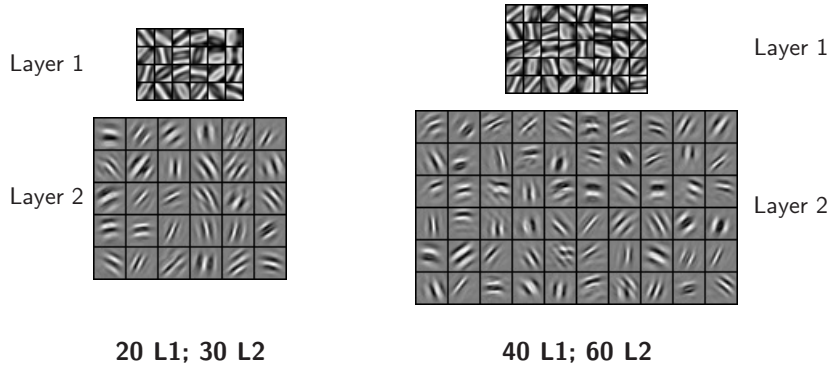

Figure 4.7 Two networks trained on the same training data with different number of weights. *left:* This network has 20 weights in layer 1, and 30 weights in layer 2, while; *right:* has 40 weights in layer 1, and 60 weights in layer 2.

It can be seen from Figure 4.7 that having fewer weights in the first layer restricts the number of learnt oriented ridges, and hence the network is not able to capture all orientations of fingerprint ridges. It also restricts the number of frequencies. Overall, the learnt filters will show a good response at fewer places in test images, as compared to networks that have more weights.

We perform the matching exercise on images reconstructed by these two networks too. It is observed that as the number of weights in the layers increases, the quality of reconstruction also increases significantly. The ROC curves and EERs are superior for the network with 60 weights in both layers than the other two. Further, there is a significant qualitative improvement as the number of weights is increased. Figure 4.8 shows comparative ROC curves for the three networks, with equal error rates being tabulated in Table 4.3.

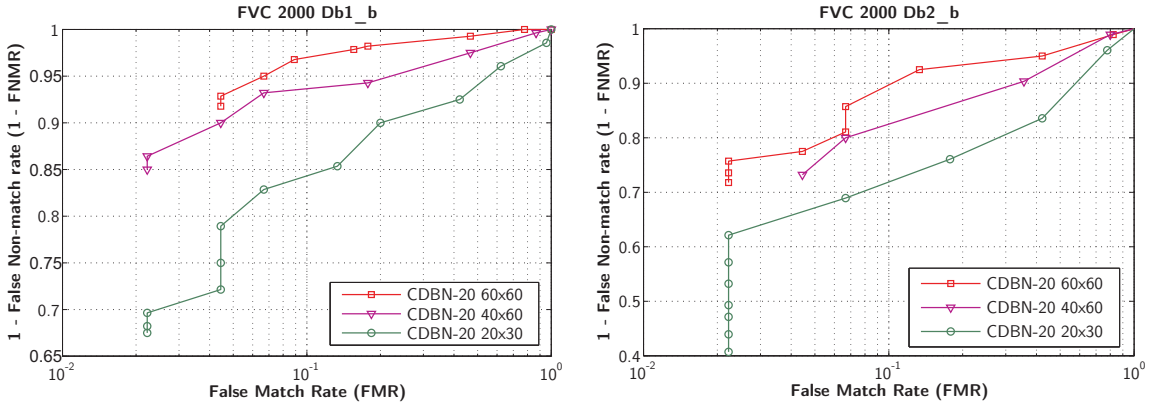

Figure 4.8 ROC curves for three networks on the FVC 2000 Db1_b (left) and FVC 2000 Db2_b (right) datasets. We see a gradual increase in performance of the convolutional DBNs as the number of weights in increased. The notation $a \times b$ denotes a network with a weights in layer 1 and b weights in layer 2.

Network	FVC 2000 Db1_a	FVC 2000 Db1_b	FVC 2000 Db2_b
60×60	6.62	5.82	10.44
40×60	8.31	6.78	16.48
20×30	11.47	14.11	22.48

Table 4.3 Equal error rates (in percentage) for three different networks on two datasets. An improvement in EER is observed as the number of weights in layers is increased. The notation $a \times b$ denotes a network with a weights in layer 1 and b weights in layer 2.

A qualitative comparison of reconstruction of the same fingerprint using the three networks is given in Figure 4.9. As hypothesised, we find that using more weights in a layer increases the learning capacity of the convolutional deep belief network, and the quality of its reconstructions also increases.

Figure 4.9 Reconstructions of the same fingerprint using three different networks. All of the shown reconstructions are after 20 iterations of hierarchical probabilistic inference.

4.5.6 Intrinsic Images' Estimation

We can estimate the orientation field, frequency image, and region mask for an input fingerprint using the learnt weights. The estimation is done using equations described in Section 4.4.5. We only require the visible and hidden units in the first layer to compute the intrinsic images. Figure 4.10 shows

the estimates of orientation field, frequency image, and region mask for a fingerprint. We also see how

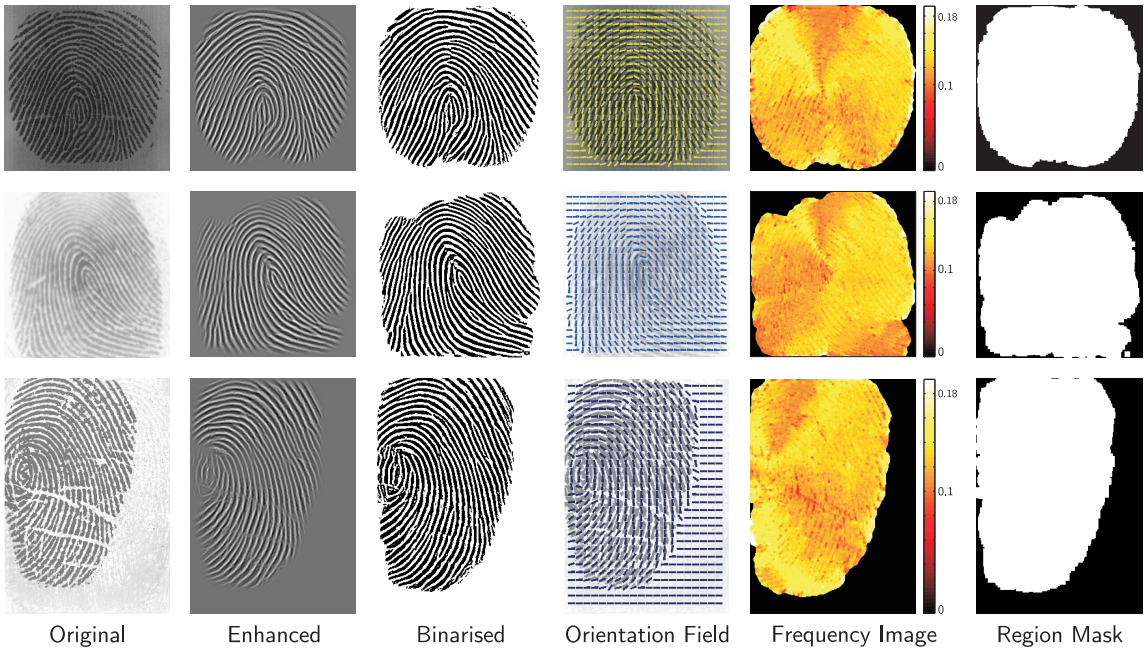

Figure 4.10 Intrinsic images inferred using the convolutional DBN. From left to right, original image, enhanced image, binarised image, orientation field, frequency image, and region mask. Each row shows these intrinsic images generated from the left-most image of the row.

the reconstruction is affected by the presence of more layers in the convolutional DBN, hence giving better estimates of orientation field, frequency image, and region mask.

Of particular importance here is the region mask obtained from the reconstructions. Several fingerprint segmentation algorithms used currently work by dividing the images into blocks. However, using a trained convolutional deep belief network, we are able to find the region mask with a precision of one pixel.

The second fingerprint in Figure 4.10 is a case where region mask estimation doesn't work very well. As can be seen, the top-most region of the fingerprint has been excluded from the mask. This was observed particularly in the dataset that this image has been taken from (FVC 2000 Db1_b), because the images in this dataset have a significantly low contrast. The ridges and valleys, hence, become inseparable in some regions. Consequently, the convolutional deep belief network, when calculating the inference at the first layer, doesn't detect a fingerprint structure at these points, and hence it is is not reconstructed.

4.6 Summary

In this chapter, we presented the use of convolutional deep belief network for unsupervised feature learning of fingerprint images. As convolutional deep belief networks are generative models, we

are able to perform reconstructions of fingerprint images using what the network learns. These reconstructions serve as enhancements of these images. Due to the convolutional nature of the CDBN, which goes hand-in-hand with the traditional contextual filtering approach applied for fingerprint enhancement so extensively, the reconstructions using a learnt network are also in-line with Gabor- and Fourier transform-based enhancements. We have also shown that adding higher layers to our network significantly improves the quality of enhancement. Experiments showed that the matching accuracy on enhanced images was better than Gabor-based enhancement on three datasets.

Future work on this path includes adding more layers to the network so that extremely noisy fingerprints can also be recovered. Adding more layers gives us greater hierarchical inference which can be used to extrapolate orientations for noisy regions. We also aim at developing a scalable model on an efficient architecture which can work on huge training data sets, and can learn multiple layers of features.

Chapter 5

Conclusions

In this thesis, we have proposed the use of unsupervised feature learning and generative neural networks to enhance fingerprint structures. On the way to doing so, we saw a preliminary use of learning to perform enhancement: the use of continuous restricted Boltzmann machines to represent patterns in local orientation fields of fingerprint images. Restricted Boltzmann machines are capable of learning patterns in training data, and then "seeing" the patterns learnt by them in the given input. This is evident from the fact that if an RBM is trained on a certain pattern, and it is then given a random input, the output it produces is an effort to locate or extract the learnt patterns in the random input.

In this thesis, we have presented an alternative approach to attacking the problem of fingerprint image enhancement. We have presented algorithms to incorporate neural networks for unsupervised feature learning of fingerprint structures. We used the continuous restricted Boltzmann machine to learn patterns in local orientation fields, which were reflected in the weights learnt by the network. The continuous RBMs were able to perform noise removal while reconstructing orientation fields of a fingerprint. However, this required a preliminary estimate of the orientation field taken by the gradient-based method. To eliminate any steps which require pre-computation, and move towards working directly on greyscale images, we used convolutional deep belief networks to learn ridge structures in fingerprints. There were several advantages of working on greyscale images using a convolutional network:

1. Scalability: We could now work on complete fingerprint images instead of working on patches, with the learning on complete images being feasible with respect to time.

2. The convolutional nature: Popular fingerprint enhancement approaches include applying contextual filters to fingerprint images. Such filters are designed so that they perform low-pass filtering along the direction of ridges, and band-pass filtering perpendicular to ridges (Section 2.2). The nature of the features learnt by a convolutional DBN is such that these motives are achieved.

3. No pre-computation: We don't need preliminary feature extraction to train the convolutional network, unlike the continuous RBMs where we had to extract orientation fields.

4. Hierarchical probabilistic inference works well with point 1. to enable higher-level data to "fill-in" regions in the enhanced fingerprint, which were too noisy to be inferred from the first layer only.

5. One learnt model can be used to estimate several properties of a fingerprint image. For example, orientation fields, frequency images and region masks were successfully generated solely from greyscale fingerprint images using a learnt network, as shown in Section 4.5.6.

We hope that the exploration presented in this thesis can serve as the spawn of a new direction of research on the subject. We have reason to believe that deep learning can involve itself more in the task of fingerprint image enhancement. Recent successes in deep learning have shown that there lie vast and unexplored territories ahead of us. Several problems that have been attacked with training of deep neural networks in mind have been able to beat existing state-of-the-art methods for that problem. Although we might be quite a distance from reaching that mark, we can expect a strong and robust network when scaled up to include many training images, more layers and higher number of weights per layer.

Convolutional DBNs were shown to learn sparse features even from training data that had several classes in it. For instance, a network trained on images of faces, cows, chairs, and elephants was able to learn higher-level features that did not mix lower level features of two different classes. Sparse feature learning in the higher layers was seen to perform well. Extending this to fingerprints, a vast diversity in the training data, in terms of print type, changes in the orientation and frequency of ridges, difference in classes of fingerprint images, and variations in the contrast and pixel intensity of the training images can be tolerated by the convolutional DBN. It should be fruitful, then, to analyse and evaluate the results of scaling of the training data.

Furthermore, the generative nature of the convolutional DBN enables us to reconstruct images from learnt representations. Thinking along the lines of elementary deep belief networks, which are able to generate a belief from only the representation, a possible direction of further research can be into synthetic fingerprint generation using convolutional DBNs. For instance, setting certain hidden neurons in the highest layers of the CDBN to fire, and propagating the data downward into the network can generate a fingerprint from the representation that was set only in the highest layer.

We also saw that a convolutional DBN trained on fingerprint images estimates the region mask. We estimated the region mask using a trained network by looking at places in a given image where the network performs any reconstruction. Since the CDBN has learnt only fingerprint patterns, any reconstruction by the network is indicative of the presence of a fingerprint pattern. We might extend this line of thinking to detection of minutiae in a fingerprint. Given a network trained on minutiae patterns, it should be interesting to note what kind of detection and/or localisation the network is able to make.

The future work using this as a starting point is vast. Only by going deeper into it, shall we find out what impact it might have on current fingerprint recognition systems and tasks.

Appendix A

Gabor Filters

A.1 Equation

A Gabor filter is a filter in the spatial domain and can be presented as a 2-dimensional Gaussian surface, with a cosine decay along one of the directions. More formally, it is the product of a 2-dimensional Gaussian of the form $\exp\left(-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}\right)$ and a cosine along a direction, $\cos(2\pi f x)$. Hence, it has the following parameters:

- **The frequency**, f. This determines the frequency of the cosine along the direction of tapering, d.
- **The orientation**, θ. The orientation determines the direction of the cosine decay, i.e., d.
- **Phase of the cosine**, ϕ. The phase shift in the cosine signal.
- **The standard deviations**, σ_1 and σ_2. The standard deviations of the Gaussian surface along the d-direction, and the direction perpendicular to it.

The equation of a Gabor filter is hence:

$$G(x, y : f, \theta, \phi, \sigma_1, \sigma_2) = \exp\left(-\frac{x'^2 + \gamma y'^2}{2\sigma^2}\right)\cos\left(2\pi x' f + \phi\right) \tag{A.1}$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$ represent rotations of the axes by θ; and $\gamma = \frac{\sigma_1}{\sigma_2}$ is the aspect ratio of the surface. Figure A.1 shows two representations of the filter.

If we compare the ridge structure at point in a fingerprint with a Gabor filter designed with the appropriate orientation and frequency (which would be dictated by the ridge orientation and frequency at that point), we find that a strong correlation between them. This filters the image and what remains is an almost ideal ridge structure, which resembles the Gabor filter. As the Gabor filter takes only one orientation, we have to consider a small-enough region so that that the orientation of ridges can be assumed constant. Further, the decay along both directions of the Gabor filter is consistent with this assumption.

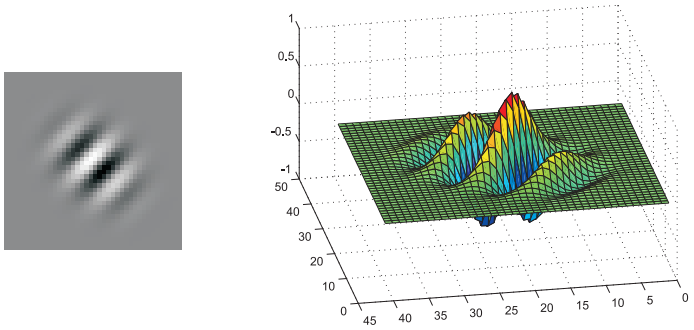

Figure A.1 Two representations of a Gabor filter with $\theta = \pi/4$, $f = 0.125$, $\phi = \pi/3$, $\sigma_1 = 6$, $\gamma = 2$, and $\sigma_2 = \sigma_1/\gamma = 3$. On the left is a 2-dimensional plot. To visualise this, grey pixels can be assumed to be zero, white pixels to be positive, and black pixels to be negative. On the right is a 3-dimensional surf plot.

It is interesting to note that the values of σ_1 and σ_2 play a crucial role in introducing spurious minutiae after enhancement. [22] noted that reducing σ_1 with respect to σ_2 makes the enhancement more robust to noise, and also reduces spurious ridges. This happens because reducing σ_1 results in better tolerance in frequency errors. [66] proposed increasing σ_2 instead near the singularities.

Appendix B

Block Gibbs Sampling in Hierarchical Probabilistic Inference

Hierarchical probabilistic inference, introduced in Chapter 4, is an algorithm to draw an inference using the higher layers of a convolutional deep belief network, which learn high-level features, and combine this top-down inference with bottom-up inference to generate a reconstructed image which is governed by low-level as well as high-level features. It was mentioned in that chapter that we used block Gibbs sampling in hierarchical probabilistic inference. Block Gibbs sampling says that when drawing from probability distributions, two or more variables are clubbed together (which are otherwise independent) to infer the rest of the variables. Next, the remaining variables are clubbed together to infer the first set. An instance of this was observed in restricted Boltzmann machines (RBMs), where we used visible units to calculate the activations of hidden units, and vice versa. However, it gets complicated with a convolutional deep belief network. As all units in all layers are considered to have a joint probability distribution, clubbing variables together in this scenario needs to be done carefully. We will discuss two cases in this appendix: one where the network has two layers, and another where it has three.

Recall from Chapter 4 that we use the visible units of a layer l, and the hidden units of the next layer, $l+1$, to calculate activations of hidden and pooling neurons in layer l. Call V_1, the set of visible neurons in the first layer. Similarly, call H_k and P_k, the set of hidden and pooling neurons, respectively, in the k-th layer. Then, hierarchical probabilistic inference says that we combine H_{k+1} and P_{k-1} to infer H_k and P_k. To simplify, we group the units of every alternate layer. Hence, units of the first layer, third layer, fifth layer, and so on, shall be put into one set of units; and units of the second layer, fourth layer, sixth layer, and so on, into the other set.

B.1 Two-layered network

For a two-layered network, as mentioned in the previous section, we draw H_1 and P_1, conditioned on V_1 and H_2 (there are no pooling units before H_1, hence there's no P_0. V_1 is used instead). From P_1, we now calculate H_2, using the weights in the second layer. We can continue by drawing H_1 and P_1, again conditioned on V_1 and H_2. Figure B.1 illustrates this process.

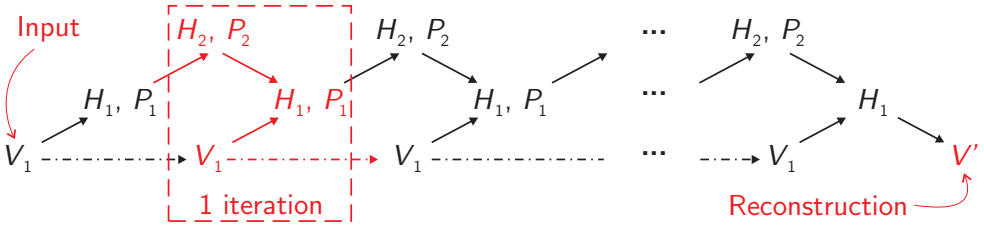

Figure B.1 Hierarchical probabilistic inference using two layers. All variables are divided into two sets: $\{V_1, H_2, P_2\}$ and $\{H_1, P_1\}$. At every iteration, each set of variables is computed using the other set. The solid arrows indicate that the variables at the end of the arrows were computed from the variables at the bases. The dotted arrows indicate that the variables weren't computed, but only used in the next calculation.

B.2 Three-layered network

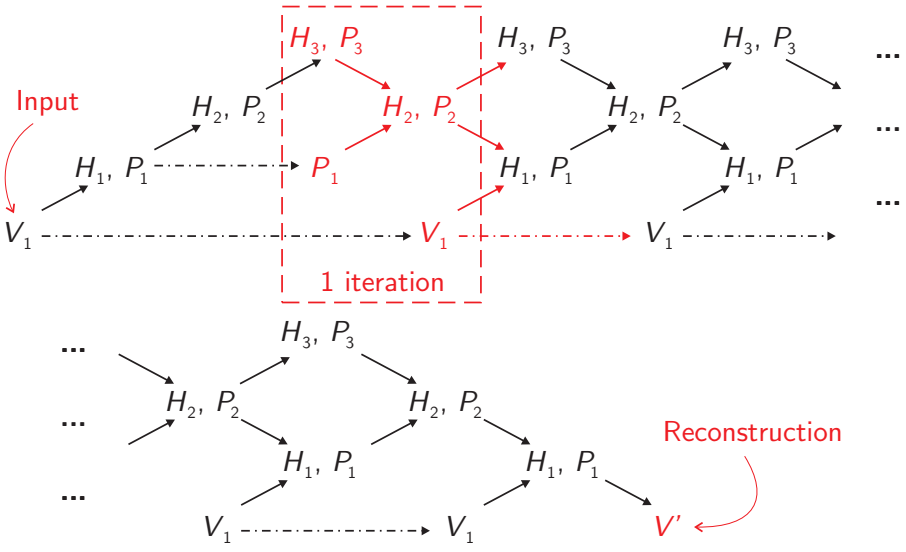

Figure B.2 Hierarchical probabilistic inference using three layers. All variables are divided into two sets: $\{V_1, H_2, P_2\}$ and $\{H_1, P_1, H_3, P_3\}$. At every iteration, each set of variables is computed using the other set. The solid arrows indicate that the variables at the end of the arrows were computed from the variables at the bases. The dotted arrows indicate that the variables weren't computed, but only used in the next calculation.

Again, we need to group variables into two sets. We first traverse all the layers, and generate H_3. Next, we calculate H_2, which is given by H_3 and P_1. We now use H_2 to generate H_3 and P_3, and compute H_1 and P_1 from P_2 and V_1. Figure B.2 illustrates this process.

B.3 A Note on the Sampling Process

It is important to note that even though V_1 can be inferred from H_1 (solely) at each iteration (in both, a 2-layered and a 3-layered network), we do this only at the last iteration, which gives us the final reconstruction of the input image. Intuitively, this can be thought of as trapping high level data in the higher layers, and using top-down inference to push the effects of this high level learning onto the first layer. Empirically too, this modification gives far better results than using the reconstructed visible units at every iteration. Another way to interpret this is asking the network to form a combined inference form top-down data and the input image.

Related Publications

1. *Learning Fingerprint Orientation Fields Using Continuous Restricted Boltzmann Machines*
   Mihir Sahasrabudhe and Anoop M. Namboodiri, 2nd Asian Conference on Pattern Recognition, 5-8 November 2013, Okinawa, Japan.

2. *Fingerprint Enhancement Using Unsupervised Hierarchical Feature Learning*
   Mihir Sahasrabudhe and Anoop M. Namboodiri, 9th Indian Conference on Vision, Graphics and Image Processing, 14-18 December 2014, Bangalore, India. **(Oral)**

Bibliography

[1] A. Almansa and T. Lindeberg. Fingerprint enhancement by shape adaptation of scale-space operators with automatic scale selection, 2000.

[2] A. A. Altun and N. Allahverdi. Neural network based recognition by using genetic algorithm for feature selection of enhanced fingerprints. In *Adaptive and Natural Computing Algorithms*, pages 467–476. Springer, 2007.

[3] D. Arpit and A. M. Namboodiri. Fingerprint feature extraction from gray scale images by ridge tracing. In *IJCB*, pages 1–8, 2011.

[4] A. M. Bazen, G. T. B. Verwaaijen, S. H. Gerez, L. P. J. Veelenturf, and B. J. van der Zwaag. A correlation-based fingerprint verification system, 2000.

[5] S. Bernard, N. Boujemaa, D. Vitale, and C. Bricot. Fingerprint segmentation using the phase of multiscale gabor wavelets. In *The 5th Asian Conference on Computer Vision, Melbourne, Australia*. Citeseer, 2002.

[6] J. D. Bowen. The home office automatic fingerprint pattern classification project. In *Neural Networks for Image Processing Applications, IEE Colloquium on*, pages 1/1–1/5, Oct 1992.

[7] R. Cappelli, M. Ferrara, and D. Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2128–2141, 2010.

[8] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning, 2005.

[9] C. Carvalho and H. Yehia. Fingerprint alignment using line segments. In *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, pages 380–386. 2004.

[10] H. Chen and A. F. Murray. Continuous restricted boltzmann machine with an implementable training algorithm. In *Vision, Image and Signal Processing, IEE Proceedings, Volume 150, No. 3, 10.1049/ip-vis:20030362*, 2003.

[11] J. Chen, F. Chan, and Y.-S. Moon. Fingerprint matching with minutiae quality score. In *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*. 2007.

[12] H. Cheng, J. Tian, and T. Zhang. Fingerprint enhancement with dyadic scale-space. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 200–203 vol.1, 2002.

[13] K. Chih Lee and S. Prabhakar. Probabilistic orientation field estimation for fingerprint enhancement and verification. In *Biometrics Symposium, 2008. BSYM '08*, pages 41–46, Sept 2008.

[14] S. Chikkerur, A. N. Cartwright, and V. Govindaraju. Fingerprint enhancement using stft analysis. *Pattern Recognition*, 40(1):198–211, 2007.

[15] A. Crouzil, L. Massip-Pailhes, and S. Castan. A new correlation criterion based on gradient fields similarity. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 632–636 vol.1, Aug 1996.

[16] S. Dass. Markov random field models for directional field and singularity extraction in fingerprint images. *Image Processing, IEEE Transactions on*, 13(10):1358–1367, 2004.

[17] G. Desjardins and Y. Bengio. Empirical evaluation of convolutional rbms for vision. *DIRO, Université de Montréal*, 2008.

[18] A. Erol, U. Halici, and G. Ongun. Intelligent biometric techniques in fingerprint and face recognition. chapter Feature Selective Filtering for Ridge Extraction, pages 195–215. 1999.

[19] J. Feng, J. Zhou, and A. Jain. Orientation field estimation for latent fingerprint enhancement. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(4):925–940, April 2013.

[20] F. Galton. *Finger Prints*. Macmillan, London, 1892.

[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[22] S. Greenberg, M. Aladjem, D. Kogan, and I. Dimitrov. Fingerprint image enhancement using filtering techniques. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 322–325. IEEE, 2000.

[23] T. Hatano, T. Adachi, S. Shigematsu, H. Morimura, S. Onishi, Y. Okazaki, and H. Kyuragi. A fingerprint verification algorithm using the differential matching rate. In *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3 - Volume 3*, ICPR '02, 2002.

[24] G. E. Hinton. A practical guide to training restricted boltzmann machines. August 2010.

[25] L. Hong and A. Jain. Classification of fingerprint images. In *Proceedings of the 11th Scandinavian Conference on Image Analysis, Kangerlussuaq*, pages 7–11, 1999.

[26] L. Hong, Y. Wan, and A. Jain. Fingerprint image enhancement: Algorithm and performance evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):777–789, 1998.

[27] C.-T. Hsieh, E. Lai, and Y.-C. Wang. An effective algorithm for fingerprint image enhancement based on wavelet transform. *Pattern Recognition*, 36(2):303 – 312, 2003.

[28] P. Hughes and A. D. P. Green. The use of neural networks for fingerprint classification. In *Artificial Neural Networks, 1991., Second International Conference on*, pages 79–81, Nov 1991.

[29] A. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(4):348–359, Apr 1999.

[30] T.-Y. Jea and V. Govindaraju. A minutia-based partial fingerprint recognition system. *Pattern Recogn.*, 38(10), Oct. 2005.

[31] T. Kamei. Image filter design for fingerprint enhancement. In *Automatic Fingerprint Recognition Systems*, pages 113–126. Springer New York, 2004.

[32] T. Kamei and M. Mizoguchi. Image filter design for fingerprint enhancement. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 109–114, Nov 1995.

[33] K. Karu and A. K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[34] M. Kass and A. Witkin. Analyzing oriented patterns. *Comput. Vision Graph. Image Process.*, 37(3):362–385, Mar. 1987.

[35] M. Kawagoe and A. Tojo. Fingerprint pattern classification. *Pattern Recogn.*, 17(3):295–303, June 1984.

[36] M. Kayaoglu, B. Topcu, and U. Uludag. Standard fingerprint databases: Manual minutiae labeling and matcher performance analyses. *CoRR*, abs/1305.1443, 2013.

[37] B.-G. Kim and D.-J. Park. Adaptive image normalisation based on block processing for enhancement of fingerprint image. *Electronics Letters*, 38(14):696–698, Jul 2002.

[38] T. Kobayashi. A fingerprint image recognition method for network user identification. In *Computing and Information, 1992. Proceedings. ICCI '92., Fourth International Conference on*, pages 369–372, May 1992.

[39] Z. M. Kovács-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11), Nov. 2000.

[40] P. Kovesi. www.csse.uwa.edu.au/ pk/research/MatlabFns/.

[41] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[42] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[43] M.-T. Leung, W. Engeler, and P. Frank. Fingerprint image processing using neural networks. In *Computer and Communication Systems, 1990. IEEE TENCON'90., 1990 IEEE Region 10 Conference on*, pages 582–586. IEEE, 1990.

[44] D. Maio and D. Maltoni. Neural network based minutiae filtering in fingerprints. In *Pattern Recognition, 1998. Proceedings. 14th International Conference on*, pages 1654–1658, 8 1998.

[45] D. Maio, D. Maltoni, D. Cappelli, J. L. Wayman, and A. K. Jain. Fvc2000: Fingerprint verification competition, August 2000.

[46] D. Maio, D. Maltoni, D. Cappelli, J. L. Wayman, and A. K. Jain. Fvc2002: Second fingerprint verification competition. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, 2002.

[47] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer, 2009.

[48] P. Melin, D. Bravo, and O. Castillo. Fingerprint recognition using modular neural networks and fuzzy integrals for response integration. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 4, pages 2589–2594 vol. 4, July 2005.

[49] V. Mnih. http://code.google.com/p/cudamat/, 2009.

[50] NIST. NIST biometric image software. `http://nist.gov/itl/iad/ig/nbis.cfm`.

[51] L. O'Gorman and J. Nickerson. Matched filter design for fingerprint image enhancement. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 916–919 vol.2, Apr 1988.

[52] L. O'Gorman and J. V. Nickerson. An approach to fingerprint filter design. *Pattern Recogn.*, 22(1):29–38, Jan. 1989.

[53] L. O'Gorman and J. V. Nickerson. An approach to fingerprint filter design. *Pattern Recognition*, 22(1):29–38, 1989.

[54] M. Oliveira and N. Leite. A multiscale directional operator and morphological tools for reconnecting broken ridges in fingerprint images. *Pattern Recognition*, 41(1):367–377, 2008.

[55] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325, 1997.

[56] A. C. Pais Barreto Marques and A. C. Gay Thome. A neural network fingerprint segmentation method. In *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*, pages 6–pp. IEEE, 2005.

[57] G. Parziale and A. Niel. A fingerprint matching using minutiae triangulation. In *Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*. 2004.

[58] C. Quek, K. B. Tan, and V. K. Sagar. Pseudo-outer product based fuzzy neural network fingerprint verification system. *Neural Netw.*, 14(3):305–323, Apr. 2001.

[59] A. Rahman Mohamed and G. E. Hinton. Phone recognition using restricted boltzmann machines. In *ICASSP*, pages 4354–4357, 2010.

[60] R. K. N. V. Rama and A. M. Namboodiri. Fingerprint enhancement using hierarchical markov random fields. In *Proceedings of the 2011 International Joint Conference on Biometrics*, Washington, DC, USA, 2011.

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

[62] R. Salakhutdinov and G. Hinton. Replicated softmax: an undirected topic model. In *In Advances in Neural Information Processing Systems*, page 2010.

[63] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, 2007.

[64] B. Sherlock. Computer enhancement and modeling of fingerprint images. In *Automatic Fingerprint Recognition Systems*, pages 87–112. Springer New York, 2004.

[65] B. G. Sherlock, D. Monro, and K. Millard. Algorithm for enhancing fingerprint images. *Electronic Letters*, 28(18):1720, 1992.

[66] B. G. Sherlock, D. Monro, and K. Millard. Fingerprint enhancement by directional fourier filtering. *Vision, Image and Signal Processing, IEE Proceedings -*, 141(2):87–94, Apr 1994.

[67] Z. Shi and V. Govindaraju. Fingerprint image enhancement based on skin profile approximation. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 714–717, 2006.

[68] S. Sjogaard. Discrete neural networks and fingerprint identification. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 316–322, Aug 1992.

[69] P. Sutthiwichaiporn, V. Areekul, and S. Jirachaweng. Iterative fingerprint enhancement with matched filtering and quality diffusion in spatial-frequency domain. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, pages 1257–1260, 2010.

[70] G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, page 2007. MIT Press, 2006.

[71] J. Ton and A. Jain. Registering landsat images by point matching. *Geoscience and Remote Sensing, IEEE Transactions on*, 27(5):642–651, Sep 1989.

[72] R. Udupa, G. Garg, and P. Sharma. Fast and accurate fingerprint verification. In *Proc. Int. Conf. on Audio- and Video-Based biometric Person Authentication (3rd)*, pages 192–197, 2001.

[73] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[74] L. Wang and M. Dai. Application of a new type of singular points in fingerprint classification. *Pattern Recogn. Lett.*, 28(13), Oct. 2007.

[75] W. Wang, J. Li, and W. Chen. Fingerprint minutiae matching based on coordinate system bank and global optimum alignment. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 401–404, 2006.

[76] C. I. Watson, G. Candela, and P. Grother. Comparison of fft fingerprint filtering methods for neural network classification. *NISTIR*, 5493, 1994.

[77] C. I. Watson and C. L. Wilson. NIST special database 4, fingerprint database, March 1992.

[78] D. Weber. A cost effective fingerprint verification algorithm for commercial applications. In *Communications and Signal Processing, 1992. COMSIG '92., Proceedings of the 1992 South African Symposium on*, pages 99–104, Sep 1992.

[79] A. Willis and L. Myers. A cost-effective fingerprint recognition system for use with low-quality prints and damaged fingertips. *Pattern Recognition*, 34(2):255 – 270, 2001.

[80] C. L. Wilson, G. T. Candela, and C. I. Watson. Neural network fingerprint classification, 1998.

[81] C. Wu and V. Govindaraju. Singularity preserving fingerprint image adaptive filtering. In *Image Processing, 2006 IEEE International Conference on*, pages 313–316, Oct 2006.

[82] N. Yager and A. Amin. Fingerprint alignment using a two stage optimization. *Pattern Recogn. Lett.*, 27(5):317–324, Apr. 2006.

[83] H. Yahagi, S. Igaki, and F. Yamagishi. Moving-window algorithm for fast fingerprint verification. In *Southeastcon '90. Proceedings., IEEE*, pages 343–348 vol.1, Apr 1990.

[84] J. Yang, L. Liu, T. Jiang, and Y. Fan. A modified gabor filter design method for fingerprint image enhancement. *Pattern Recognition Letters*, 24(12):1805–1817, 2003.

[85] Q. Zhang and H. Yan. Fingerprint classification based on extraction and analysis of singularities and pseudo ridges. *Pattern Recognition*, 37(11):2233 – 2243, 2004.

[86] E. Zhu, J. Yin, and G. Zhang. Fingerprint enhancement using circular gabor filter. In *Image Analysis and Recognition*, pages 750–758. Springer, 2004.