Communicated by Richard Lippmann

Neural Networks for Fingerprint Recognition

Pierre Baldi
Jet Propulsion Laboratory and Division of Biology, California Institute of Technology, Pasadena, CA 91109 USA

Yves Chauvin
Net-ID, Inc. and Department of Psychology, Stanford University, Stanford, CA 94305 USA

After collecting a data base of fingerprint images, we design a neural network algorithm for fingerprint recognition. When presented with a pair of fingerprint images, the algorithm outputs an estimate of the probability that the two images originate from the same finger. In one experiment, the neural network is trained using a few hundred pairs of images and its performance is subsequently tested using several thousand pairs of images originated from a subset of the data base corresponding to 20 individuals. The error rate currently achieved is less than 0.5%. Additional results, extensions, and possible applications are also briefly discussed.

1 Introduction

The fast, reliable, and computerized classification and matching of fingerprint images is a remarkable problem in pattern recognition that has not, to this date, received a complete solution. Automated fingerprint recognition systems could in principle have an extremely wide range of applications, well beyond the traditional domains of criminal justice and, for instance, render the use of locks and identification cards obsolete. Our purpose here is to give a brief account of our preliminary results on the application of neural network ideas to the problem of fingerprint matching. In particular, we shall describe the architecture, training, and testing of a neural network algorithm that, when presented with two fingerprint images, outputs a probability p that the two images originate from the same finger.

There are several reasons to suspect that neural network approaches may be remarkably well suited for fingerprint problems. First, fingerprints form a very specific class of patterns with very peculiar flavor and statistical characteristics. Thus the corresponding pattern recognition problems seem well confined and constrained, perhaps even more so than in other pattern recognition problems, such as the recognition of handwritten characters, where neural networks have already been applied with reasonable success (see, for instance, Le Cun *et al.* 1990).

Second, neural networks could avoid some of the pitfalls inherent to other more conventional approaches. It has been known for over a century (see Moenssens 1971 for an interesting summary) that pairs of fingerprint images can be matched by human operators on the basis of minutia and/or ridge orientations. Minutia are particular types of discontinuities in the ridge patterns, such as bifurcations, islands, and endings. There is typically of the order of 50 to 150 minutia (Fig. 2a) on a complete fingerprint image. Ten matching minutia or so are usually estimated as sufficient to reliably establish identity. Indeed, it is this strategy based on minutia detection and matching that has been adopted in most of the previous attempts to find automated solutions. The minutia-based approach has two obvious weaknesses: it is sensitive to noise (especially with inked fingerprints, small perturbations can create artificial minutia or disguise existing ones) and computationally expensive since it is essentially a graph matching problem.

Third, neural networks are robust, adaptive, and trainable from examples. This is particularly important since fingerprint images can include several different sources of deformation and noise ranging from the fingers and their positioning on the collection device (translation, roll, rotation, pressure, skin condition) to the collection device itself (ink/optical). Furthermore, it is important to observe that the requirements in terms of speed, computing power, probability of false acceptance and false rejection, memory and data base size can vary considerably depending on the application considered. To access a private residence or private car, one needs a small economic system with a small modifiable data base of a few people and a response time of at most a few seconds. On the other hand, forensic applications can require rapid searches through very large data bases of millions of records using large computers and a response time that can be longer. Neural networks can be tailored and trained differently to fit the particular requirement of specific applications.

From a technical standpoint, there are two different problems in the area of fingerprint analysis: classification and matching. The classification of fingerprints into subclasses can be useful to speed up searches through large data bases. It is of interest to ask whether neural networks can be used to implement some of the conventional classification schemes, such as the partition of fingerprints patterns into whorls, arches, and loops ("pattern level classification"), or to create new classifications boundaries. Classification problems, however, will be discussed elsewhere. Here, we shall exclusively concentrate on the matching problem. Indeed, at the core of any automated fingerprint system, whether for identification or verification purposes and whether for large or small data base environments, there should be a structure that, when presented with two fingerprint images, decides whether or not they originate from the same finger. Accordingly, our goal is going to be the design and testing of such a neural algorithm.

Because neural networks are essentially adaptive and need to be trained from examples, we next describe our data base of training and testing examples and how it was constructed. We then consider the matching algorithm that consists of two stages: a preprocessing stage and a decision stage. The preprocessing stage basically aligns the two images and extracts, from each one of them, a central region. The two central regions are fed to the decision stage, which is the proper neural network part of the algorithm and subject to training from examples. Whereas the preprocessing stage is fairly standard, the decision stage is novel and based on a neural network that implements a probabilistic Bayesian approach to the estimate of the probability p of a match. In the main experiment, the network is trained by gradient descent using a training set of 300 pairs of images coming from 5 fingers, 5 images per finger. Its performance is then validated using an additional set of 4650 pairs coming from 15 additional fingers, 5 images per finger also. After training, the network achieves an overall error rate of 0.5%. Additional results and possible extensions are discussed at the end.

2 Data Base

Although there exist worldwide many fingerprint data bases, these are generally not available for public use. In addition, and this is a crucial issue for connectionist approaches, most data bases contain only one image or template per finger whereas training a neural network to recognize fingerprint images requires that several noisy versions of the same record be available for training. Therefore, to train and test a neural network, one must first construct a data base of digitized fingerprint images.

Such images can be obtained in a variety of ways, for instance by digital scanner with inked fingerprints or by more sophisticated holographic techniques (Igaki *et al.* 1990). We decided to build our own collection device, using a simple principle. The device basically consists of a prism placed in front of a CCD (charge coupled device) camera connected to a frame grabber board installed on a personal computer (Fig. 1). When a finger is positioned on the diagonal face of the prism, incoming rays of light from one of the square sides of the prism are refracted differently depending on whether they encounter a point of contact of the finger with the prism (corresponding to a ridge) or a point of noncontact. This creates a pattern of bright and dark ridges in the refracted beam that can easily be focused, with a lens, on the CCD camera and then digitized and stored in the computer. Our resulting images are 512×464 pixels in size, with 8 bits gray scale per pixel. On the corresponding scale, the thickness of a ridge corresponds to 6 pixels or so. This is of course not a very economical format for the storage of fingerprint images that contain

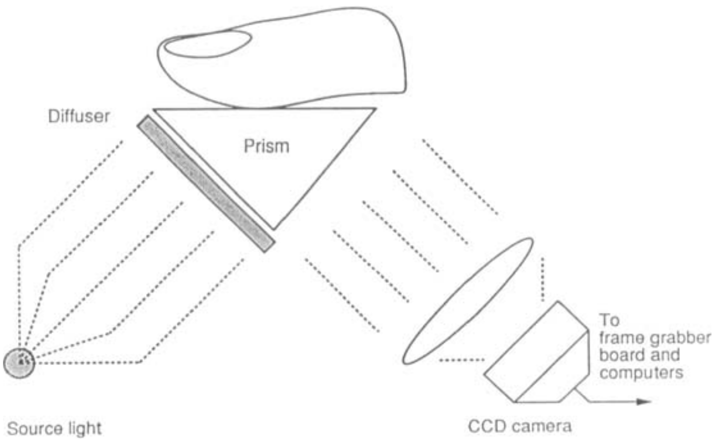

Figure 1: Collection device: diffuse light entering the prism is not reflected where the ridges are in contact with the prism. The corresponding pattern of light and dark ridges is focused on a CCD camera, digitized on a personal computer, and sent to a workstation for further processing.

a much smaller amount of useful information. Yet, this format is necessary at least in the developing phase, in particular in order to fine tune the preprocessing.

In this way, we have assembled a data base of over 200 fingerprint images using various fingers from 30 different persons. To solve the matching problem, it is imperative that the data base contains several different images of the same finger taken at different times. Thus, for what follows, the most important part of the data base consists of a subset of 100 images. These are exclusively index finger images from 20 different persons, 5 different images being taken for each index finger at different times. At each collection time, we did not give any particular instruction to the person regarding the positioning of the finger on the prism other than to do so "in a natural way." In general, we made a deliberate attempt not to try to reduce the noise and variability that would be present in a realistic environment. For instance, we did *not* clean the surface of the prism after each collection. Indeed, we do observe significant differences among images originating from the same finger. This variability results from multiple sources, mostly at the level of the finger (positioning, pressure, skin condition) and the collection device (brightness, focus, condition of prism surface).

We have conducted several learning experiments using this data base, training the networks with image pairs originated from up to 7 persons and testing the algorithm on the remaining pairs. Here, we shall mostly report the typical results of one of our largest experiments where, out of the $\binom{100}{2} = 4950$ image pairs in this data base, $\binom{25}{2} = 300$ image pairs originated from 5 different persons are used to train the network by gradient descent. The remaining 4650 pairs of images are used to test the generalization abilities of the algorithm.

Given two fingerprint images A and B, the proposition that they match (or do not match) will be denoted by $M(A, B)$ [or $\overline{M}(A, B)$]. The purpose then is to design a neural network algorithm that when presented with a pair (A, B) of fingerprint images outputs a number $p = p(M) = p[M(A, B)]$ between 0 and 1 corresponding to a degree of confidence (Cox 1946) or probability that the two fingerprints match. Here, as in the rest of the paper, we shall tend to omit in our notation the explicit dependence on the pair (A, B) except the first time a new symbol is introduced.

3 Preprocessing Stage

Any algorithm for fingerprint recognition may start with a few stages of standard preprocessing where the raw images may be rotated, translated, scaled, contrast enhanced, segmented, compressed, or convolved with some suitable filter. In our application, the purpose of the preprocessing stage is to extract from each one of the two input images a central patch called the central region and to align the two central regions. Only the two aligned central regions are in turn fed to the decision stage. The preprocessing itself consists of several steps, first to filter out high-frequency noise, then to compensate for translation effects present in the images and to segment them and finally to align and compress the central regions. For ease of description, one of the input images will be called the reference image and the other one the test image, although there is no intrinsic difference between the two.

3.1 Low Pass Filtering. To get rid of the numerous high-frequency spikes that seem to be present in the original images, we replace every pixel that significantly deviates from the values of its four neighbors by the corresponding average.

3.2 Segmentation. For each image, we first draw a tight rectangular box around each fingerprint using an edge detection algorithm and determine the geometric center of the box. The central region of the reference image is then defined to be the 65×65 central square patch that occupies the region immediately below the previously described center. For the test image, instead we select a similar but larger patch of size 105×105 (extending the previous patch by 20 pixels in each direction). This larger patch is termed the window.

3.3 Alignment. We slide, pixel by pixel, the central region of the reference image across the window of the test image (by 20 pixels up, down left and right) and compute at each step the corresponding correlation, until we find the position where the correlation is maximal. This, aside from the training period, is the most computationally expensive part of the entire algorithm. The central region of the test image is then determined by selecting the central 65×65 patch corresponding to the position of maximal correlation (Fig. 2b).

3.4 Compression and Normalization. Finally, each one of the two 65×65 central regions is reduced to a 32×32 array by discrete convolution with a truncated gaussian of size 5×5. This 32×32 compressed central region contains a low-resolution image, which corresponds roughly to 10 ridges in the original image. The resulting pixel values are conveniently normalized between 0 and 1.

In our implementation, all the parameters and in particular the size of the various rectangular boxes are adjustable. The values given here are the ones used in the following simulations and empirically seem to yield good results. To avoid border effects, a 2-pixel-wide frame is usually added around the various rectangular boxes, which explains some of the odd sizes. It is also natural to wonder at this stage whether a decision regarding the matching of the two inputs could not already be taken based solely on the value of the maximal correlation found during the alignment step (3.3) by thresholding it. It is a key empirical observation that this maximal correlation, due in part to noise effects, is *not* sufficient. In particular, the correlation of both matching and nonmatching fingerprint images is often very high (above 0.9) and we commonly observe cases where the correlation of nonmatching pairs is higher than the correlation of matching pairs. It is therefore essential to have a nonlinear decision stage following the preprocessing. Finally, it should be noticed that during training as well as testing, the preprocessing needs to be applied only once to each pair of images. In particular, only the central regions need to be cycled through the neural network during the training phase. Although the preprocessing is not subject to training, it can be implemented, for most of its part, in a parallel fashion compatible with a global neural architecture for the entire algorithm.

4 Neural Network Decision Stage

The decision stage is the proper neural network part of the algorithm. As in other related applications (see, for instance, Le Cun *et al.* 1990), the network has a pyramidal architecture, with the two aligned and compressed central regions as inputs and with a single output p. The bottom level of the pyramid corresponds to a convolution of the central

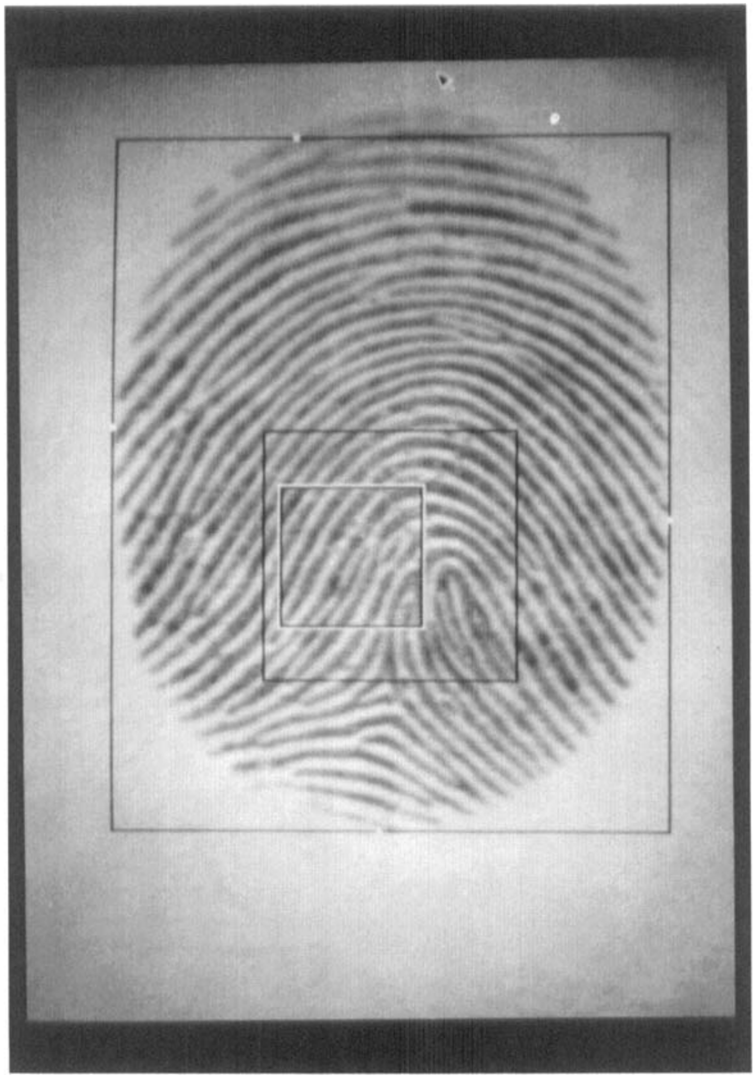

Figure 2: (a) A typical fingerprint image: the surrounding box is determined using an edge detection algorithm. Notice the numerous minutia and the noise present in the image, for instance in the form of ridge traces left on the prism by previous image collections.

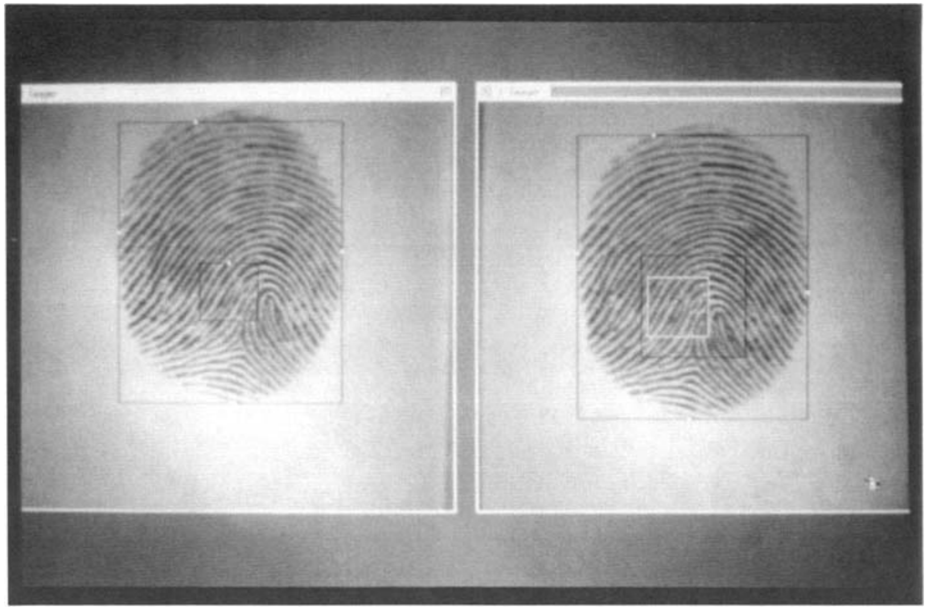

Figure 2: (b) Preprocessing of two images of the same finger: the left image is the reference image, the right image is the test image (same as a). The 65 × 65 central region of the reference image is shown in black right under the geometrical center (white dot). The 105 × 105 window of the test image is shown in black. The white square is the central region of the test image and corresponds to the 65 × 65 patch, within the window, which has a maximal correlation with the central region of the reference image.

regions with several filters or feature detectors. The subsequent layers are novel. The final decision they implement results from a probabilistic Bayesian model for the estimation of p, based on the output of the convolution filters. Both the filtering and decision part of the network are adaptable and trained simultaneously.

4.1 Convolution. The two central regions are first convolved with a set of adjustable filters. In this implementation only two different filter types are used, but the extension to a larger number of them is straightforward. Here, each filter has a 7×7 receptive field and the receptive fields of two neighboring filters of the same type have an overlap of 2 pixels to approximate a continuous convolution operation. The output of all the filters of a given type form a 6×6 array. Thus each 32×32 core is transformed into several 6×6 arrays, one for each filter type. The output of filter type j at position (x, y) in one of these arrays is given (for instance for A) by

$$z^j_{x,y}(A) = f\left(\sum_{r,s} w^j_{x,y,r,s} I_{r,s}(A) + t^j\right) \tag{4.1}$$

where $I_{r,s}(A)$ is the pixel intensity in the compressed central region of image A at the (r,s) location, f is one of the usual sigmoids $[f(x) = (1+e^{-x})^{-1}]$, $w_{x,y,r,s}$ is the weight of the connection from the (r,s) location in the compressed central region to the (x,y) location in the array of filter outputs, and t^j is a bias. The sum in 4.1 is taken over the 7×7 patch corresponding to the receptive field of the filter at the (x,y) location. The threshold and the 7×7 pattern of weights are characteristic of the filter type (so-called "weight sharing"), so that they can also be viewed as the parameters of a translation invariant convolution kernel. They are shared within an image but also across the images A and B. Thus $w^j_{x,y,r,s} = w^j(x-r, y-s)$ and, in this implementation, each filter type is characterized by $7 \times 7 + 1 = 50$ learnable parameters. In what follows, we simplify the notation for the location of the outputs of the filters by letting $(x,y) = i$. For each filter type j, we can now form an array $\Delta z^j(A,B)$ consisting of all the squared differences

$$\Delta^j_i(A,B) = [z^j_i(A) - z^j_i(B)]^2 \tag{4.2}$$

and let $\Delta z = \Delta z(A,B)$ denote the array of all $\Delta z^j_i(A,B)$ for all positions i and filter types j (Fig. 3).

4.2 Decision. The purpose of the decision part of the network is to estimate the probability $p = p[M(A,B)/\Delta z(A,B)] = p(M/\Delta z)$ of a match between A and B, given the evidence Δz provided by the convolution filters. The decision part can be viewed as a binary Bayesian classifier. There are four key ingredients to the decision network we are proposing.

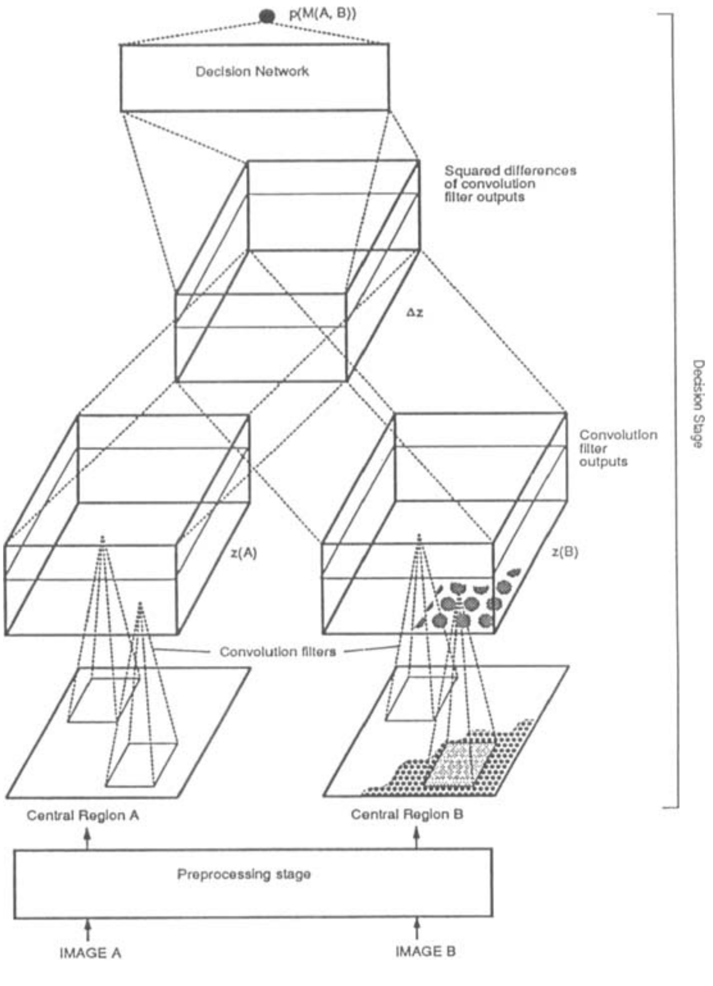

Figure 3: Network architecture: at the bottom, reference and test images A and B are presented to the network as two 32×32 arrays. The network extracts features from the images by convolution with several different 7×7 filter types.

1. Because the output of the network is to be interpreted as a probability, the usual least mean square error used in most backpropagation networks does not seem to be an appropriate measure of network performance. For probability distributions, the cross entropy between the estimated probability output p and the true probability P, summed over

all patterns, is a well-known information theoretic measure of discrepancy (see, for instance, Blahut 1987)

$$H(P,p) = \sum_{(A,B)} P \log \frac{P}{p} + Q \log \frac{Q}{q} \tag{4.3}$$

where, for each image pair, $Q = 1 - P$ and $q = 1 - p$. H is also known as the discrimination function and can be viewed as the expected value of the log-likelihood ratio of the two distributions. The discrimination is nonnegative, convex in each of its arguments, and equal to zero if and only if its arguments are equal.

2. Using Bayes inversion formula and omitting, for simplicity, the dependence on the pair (A, B)

$$p(M/\Delta z) = \frac{p(\Delta z/M)p(M)}{p(\Delta z/M)p(M) + p(\Delta z/\overline{M})p(\overline{M})} \tag{4.4}$$

The effect of the priors $p(M)$ and $p(\overline{M})$ should be irrelevant in the case of a large set of examples. Our data base is large enough for the decision to be driven only by the data, as confirmed by the simulations (see also Fig. 4). In simulations, the values chosen are typically $p(M) = 0.1$ and $p(\overline{M}) = 0.9$ [the observed value of $p(M)$ is roughly 16% in the training set and 4% in the entire data base].

3. We make the simplifying independence assumption that

$$p(\Delta z/M) = \prod_{i,j} p(\Delta z_i^j/M) \tag{4.5}$$

$$p(\Delta z/\overline{M}) = \prod_{i,j} p(\Delta z_i^j/\overline{M}) \tag{4.6}$$

Strictly speaking, this is not true, especially for neighboring locations. However, in the center of a fingerprint where there is more variability than in the periphery, it is a reasonable approximation, which leads to simple network architectures and, with hindsight, yields excellent results.

4. To completely define our network, we still need to choose a model for the conditional distributions $p(\Delta z_i^j/M)$ and $p(\Delta z_i^j/\overline{M})$. In the case of a match, the probability $p(\Delta z_i^j/M)$ should be a decreasing function of Δz_i^j. It is therefore natural to propose an exponential model of the form

$$p(\Delta z_i^j/M) = C_{ij} s_{ij}^{\Delta z_i^j}$$

where $0 < s_{ij} < 1$ and, for proper normalization, the constant C_{ij} must take the value $C_{ij} = \log s_{ij}/s_{ij} - 1$. In what follows, however, we use a less general but slightly simpler binomial model of the form

$$p(\Delta z_i^j/M) = p_j^{1-\Delta z_i^j}(1-p_j)^{\Delta z_i^j} = p_j \left(\frac{1-p_j}{p_j}\right)^{\Delta z_i^j} \tag{4.7}$$

$$p(\Delta z_i^j/\overline{M}) = q_j^{\Delta z_i^j}(1-q_j)^{1-\Delta z_i^j} = (1-q_j)\left(\frac{q_j}{1-q_j}\right)^{\Delta z_i^j} \tag{4.8}$$

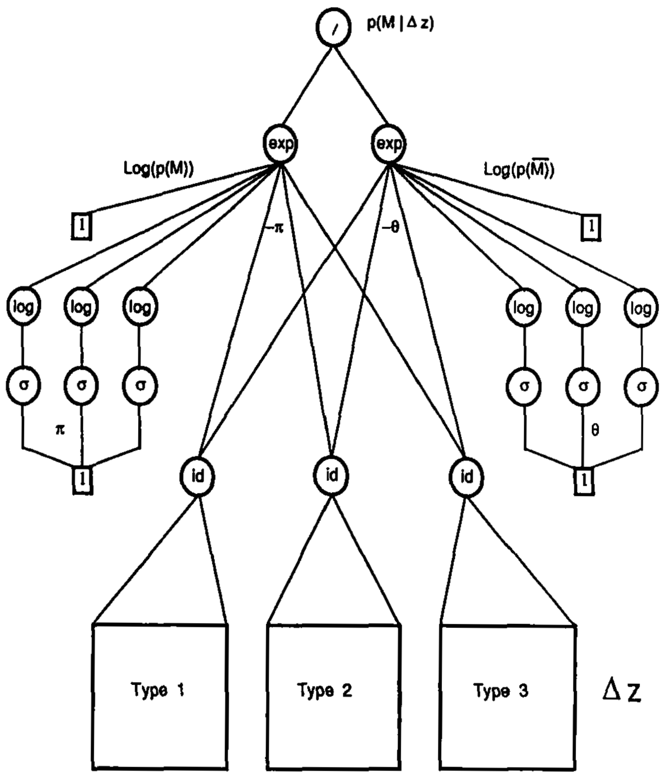

Figure 4: Neural decision network. This is just a neural network implementation of equations 4.4–4.8. Except for the output unit, each unit computes its output by applying its transfer function to the weighted sum of its inputs. In this network, different units have different transfer functions including: $\text{id}(x) = x$, $\sigma(x) = (1 + e^{-x})^{-1}$, $\log x$, and $\exp(x) = e^x$. The output unit is a normalization unit that calculates $p(M/\Delta z)$ in the form of a quotient $x/x + y$. The coefficients p_j and q_j of 4.7 and 4.8 are implemented in the form $p_j = \sigma(\pi_j)$ and $1 - q_j = \sigma(\theta_j)$. π_j and θ_j are the only adjustable weights of this part of the network and they are shared with the connections originating from the convolution filter outputs. All other connection weights are fixed to 1 except for the connections running from $\log p_j$ to the exponential unit, which have a weight equal to 36, the size of the receptive field of the convolution filters. The exponential unit on the left, for instance, computes $p(\Delta z/M)p(M) = p(M) \prod_{i,j} p_j[(1 - p_j)/p_j]^{\Delta z_i^j}$ using the fact that $(1 - p_j)/p_j = -\pi_j$. Notice that the priors play the role of a bias for the exponential units, which, after training, ends up having little influence on the output.

with $0.5 \leq p_j, q_j \leq 1$. This is again only an approximation used for its simplicity and for the fact that the feature differences Δz_i^j are usually close to 0 or 1. In this implementation and for economy of parameters, the adjustable parameters p_j and q_j depend only on the filter type, another example of weight sharing. In a more general setting, they could also depend on location.

In summary, the adjustable parameters of the neural network are $w^j{}_{x,y,r,s}$, t^j, p_j, and q_j. In this implementation, their total number is $(7 \times 7 + 1) \times 2 + 2 \times 2 = 104$. At first sight, this may seem too large in light of the fact that the network is trained using only 300 pairs of images. In reality, each one of the 50 shared parameters corresponding to the weights and bias of each one of the convolution filters is trained on a much larger set of examples since, for each pair of images, the same filter is exposed to 72 different subregions. The parameters are initialized randomly (for instance the $w^j_{x,y,r,s}$ are all drawn independently from a gaussian distribution with 0 mean and standard deviation 0.5). They are then iteratively adjusted, after each example pair presentation, by gradient descent on the cross entropy error measure H. The specific formula for adjusting each one of them can readily be derived from 4.3–4.8 and will not be given here for brevity.

It should also be noticed that the adaptable pattern classifier defined by 4.3–4.8 is not a neural network in the most restrictive sense of a layered system of sigmoid units. It is rather a nonlinear model with adjustable parameters which can be drawn (Fig. 4), and in several different ways, in a neural network fashion. The number of units and their types, however, depends on how one decides to decompose the algebraic steps involved in the computation of the final output p.

5 Results and Conclusions

We have trained the network described in the previous sections using 300 pairs of images from our data base and only two different filter types (Fig. 5). The network performance is then tested on 4650 new pairs. The network usually learns the training data base perfectly. This is not the case, however, when only one filter type is used. The separation obtained in the output unit between matching and nonmatching pairs over the entire population is good since 99% of the matching (or nonmatching) pairs yield an output above 0.8 (or below 0.2). The error rate on the generalization set is typically 0.5% with roughly half the errors due to false rejections and half to false acceptances. In many applications, these two types of error do not play symmetric roles. It is often the case, for instance, that a false acceptance is more catastrophic than a false rejection. If, by changing our decision threshold on p, we enforce a 0% rate of false acceptances, the rate of false rejections increases to 4%. This error rate

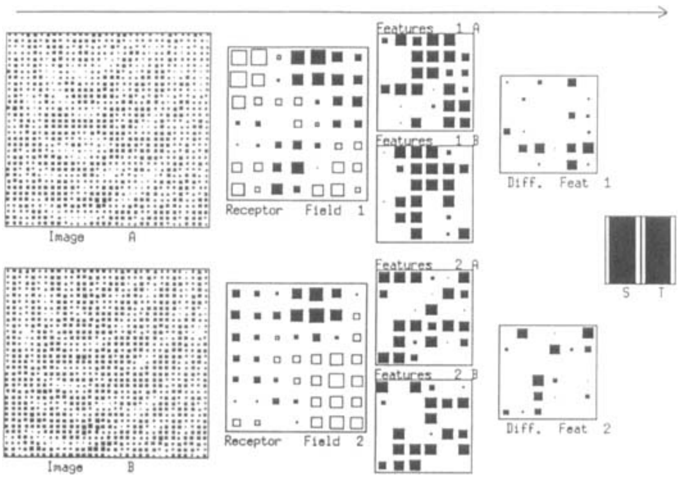

Figure 5: Unit activation throughout the network when two matching fingerprint images are presented as inputs. Flow of activation is left to right. Images A and B are presented to the network as 32×32 arrays. The network has two filters 1 and 2, each represented by the corresponding 7×7 pattern of weights. Each filter is convolved with each array and generates the set of feature arrays $1A$, $1B$, $2A$, and $2B$. The next layer computes the squared feature differences for each filter. Finally, the similarity $S = p(M/\Delta z)$ is computed. In this example the similarity is close to 1 (represented by a black vertical bar), close to the target value $T = 1$. Notice the essentially binary values assumed by the features and the compact representation of the input images with 72 bits each.

needs of course to be reduced, but even so it could be almost acceptable for certain applications.

As in other related applications, the interpretation of the filter types discovered by the network during learning is not always straightforward (Figs. 5 and 6). We have conducted various experiments with up to four filter types but on smaller data bases. Usually, at least one of the filter types always appears to be an edge or a ridge orientation detector. Some of the other filter types found in the course of various experiments may be interpretable in terms of minutia detectors, although this is probably more debatable.

On the completion of the training phase, the outputs of the filters in the decision stage are close to being binary 0 or 1. Since the final decision of the network is entirely based on these outputs, these provide a very compressed representation of all the relevant matching information originally contained in the $512 \times 464 \times 8$ bits images. Thus, in this im-

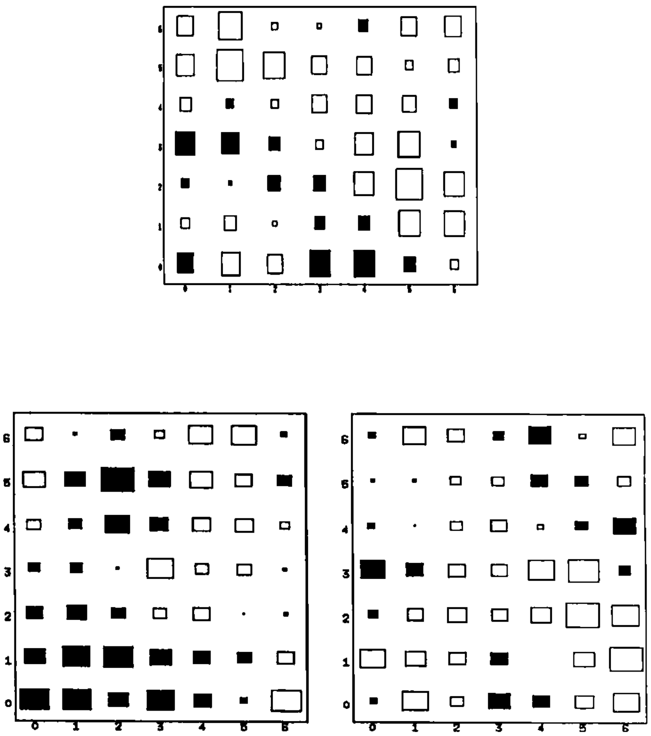

Figure 6: Examples of filters learned in different experiments: one filter is represented by the corresponding 7 × 7 patterns of weights. The size of each square represents the value of the corresponding weight. Black and white squares correspond to positive and negative weights, respectively. Whereas the first filter seems to be an edge or ridge orientation detector, the other two are more difficult to interpret. It may be tempting to describe them in terms of minutia detectors, such as an ending and a bifurcation detector, but this may not necessarily be the case.

plementation, each image is roughly reduced to 36 × 2 = 72 bits, which is within a factor of two from a rough estimate of the theoretical lower bound (the number of human beings, past and present, is approximately $2^{33} \approx 8.5 \times 10^9$).

In applications, the algorithm would not be used in the same way as it has been during the training phase. In particular, only the central regions of the reference images need to be stored in the data base. Since the forward propagation through the decision stage of the algorithm is very fast, one can in fact envision a variation on the algorithm where the alignment step in the preprocessing is modified and where the reference image is stored in the data base only in the most compressed form given by the corresponding outputs of the filters in the decision stage. In this variation, all the possible 65×65 square patches contained in the 105×105 window of the test image would be sent, after the usual compression and normalization preprocessing steps, through the convolution filter arrays and then matched through the neural network with the corresponding outputs for the reference image. The final decision would then be based on an examination of the resulting *surface* of p values. Whether this algorithm leads also to better decisions needs further investigation.

To reduce the error rate, several things can be tried. One possibility is to use more general exponential models in 4.7 and 4.8 rather than binomial distributions. Alternatively, the number of filter types or the number of free parameters could be increased (for instance by letting p_j and q_j depend also on location) as well as the size of the training and validation sets. Another possibility is to use in the comparison larger windows or, for instance, two small windows rather than one, the second window being automatically aligned by the alignment of the first one. A significant fraction of the residual false rejections we have seems to be due to rotation effects, that is, to the fact that fingers are sometimes positioned at different angles on the collection device. The network we have described seems to be able to deal with rotations up to a 10° angle. Larger rotations could easily be dealt with in the preprocessing stage. It is also possible to incorporate a guiding device into the collection system so as to entirely avoid rotation problems.

In this study, we have attempted to find a general purpose neural network matcher that could be able, in principle, to solve the matching problem over the entire population. In this regard, it is remarkable that the network, having been trained with image pairs associated with only five different persons, generalizes well to a larger data base associated with 20 persons. Obviously, a general purpose network needs also to be tested on a larger sample of the population. Unlike the classification problem, however, the matching problem should be much less sensitive to the size of the training and testing sets. In classifying whorls for instance, it is essential to expose the network to a large sample representative of whorl patterns across the entire population with all their subtle statistical variations. In our matcher, we are basically substracting one image from the other and therefore only the variability of the difference really matters.

Specific applications, especially those involving a small data base, have particular characteristics that may be advantageously exploited both in the architecture and the training of networks and raise also some particular issues. For a car lock application, for instance, positive matches occur only with a pair of fingerprints associated with a small data base of only a few persons. Positive matches corresponding to fingerprints associated with persons outside the data base are irrelevant. They may not be needed for training. Conceivably, in small data base applications, a different network could be trained to recognize each record in the data base separately against possible imposters.

Finally, the approach we have described and especially the Bayesian decision stage with its probabilistic interpretation is not particular to the problem of fingerprint recognition. It is rather a general framework that could be applied to other pattern matching problems where identity or homology needs to be established. The corresponding neural networks can easily be embodied in hardware, especially once the learning has been done off-line. As already pointed out, most of the steps in the preprocessing and the decision stages are in fact convolutions and are amenable to parallel implementation. On a workstation, it currently takes on the order of 10 sec to completely process a pair of images. This time could be reduced by a few orders of magnitude with specially dedicated hardware.

References

Blahut, R. E. 1987. *Principles and Practice of Information Theory.* Addison-Wesley, Reading, MA.

Cox, R. T. 1946. Probability, frequency and reasonable expectation. *Am. J. Phys.* **14**(1), 1–13.

Igaki, S., Eguchi, S., and Shinzaki, T. 1990. Holographic fingerprint sensor. *Fujitsu Tech. J.* **25**(4), 287–296.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. 1990. Handwritten digit recognition with a back-propagation network. In *Neural Information Processing Systems,* Vol. 2, pp. 396–404. Morgan Kaufmann, San Mateo, CA.

Moenssens, A. A. 1971. *Fingerprint Techniques.* Chilton Book Company, Radnor, PA.

Received 28 April 1992; accepted 6 October 1992.