Neural Network Fingerprint Classification

C. L. Wilson
G. T. Candela
C. I. Watson
National Institute of Standards and Technology
Advanced Systems Division
Image Recognition Group

Accepted for publication in *J. Artifical Neural Networks*,**1**, No. 2, 1993.

Abstract

A massively parallel fingerprint classification system is described that uses image-based ridge-valley features, K-L transforms, and neural networks to perform pattern level classification. The speed of classification is 2.65 seconds per fingerprint on a massively parallel computer. The system is capable of 95% classification accuracy with 10% rejects. All tests were performed using a sample of 4000 fingerprints, 2000 matched pairs. Finding two ridge-valley direction sets takes 0.5 seconds per image, the alignment 0.1 seconds per image, the K-L transform 20 ms per image, and the classification 1 ms per image. The image processing prior to classification takes more than 99% of total processing time; the classification time is 0.03% of the total system's time.

1 Introduction

1.1 Systems Description

Neural networks have had the potential for massively parallel implementation for some time, but system level image-based applications employing neural networks have only recently been realized, because the requirements for an image system include the image isolation, segmentation, and feature extraction as well as recognition.

The problem, Pattern level Classification Automation (PCA), is one of accurately classifying fingerprints into one of five classes from a fingerprint image. The five classes are arch, left loop, right loop, tented arch, and whorl. These classes are abbreviated A, L, R, T, and W in this report. A detailed description of these classes is found in [1]. This paper discusses machine learning based methods of solving the PCA problem using example images for training. The images used are 512 by 512 8-bit gray with a resolution of 20 pixels/mm.

Two neural-network-based methods are used in this system one for feature extraction and one for classification. The K-L method is used [2] for feature extraction. This is a self-organizing method [3] in that it uses no class information to select features. The K-L method maximizes the variance in a feature set by using the principal eigenfunctions of the covariance matrix of the feature set. In the fingerprint system, local ridge directions are extracted from the image and used in subsequent processing. A similar technique has also been used with wavelets for face recognition [4] and for Kanji character recognition [5]. The K-L transform is dimension reducing; for ridge-valley features, the 1680 direction components are converted to (at most) 80 features.

The features generated by the K-L transform are used for training a Multi-Layer Perceptron (MLP) using Scaled Conjugate Gradient, SCG, optimization [6]. For the problems presented here, this method

is from 10 to 100 times faster than backpropagation in training. Details of the method and procedures for obtaining the program are available in [6]. Typical classification accuracies on samples with equal numbers of each class are from 84% to 88%. When accuracies are estimated taking into account the observed a priori probabilities, accuracies of 94% have been achieved with 10% rejects; the more common types of fingerprints are easier to recognize.

1.2 Parallel SIMD Hardware Architecture

The Single Instruction Multiple Data (SIMD) architecture used for this study was an Active Memory Technology 510c Distributed Array Processor with 8-bit math coprocessors.[1] This machine consists of a 32 by 32 grid of 1-bit processor elements (PE) and a 32 by 32 grid of 8-bit processors. Operation of the PE array is controlled by a four million-instruction-per-second RISC master control unit (MCU). All program instructions are stored in a separate program memory and are passed to the PE array through the MCU. A block diagram of this architecture is shown in figure 1.

All data are stored in a separate array memory. The array memory is organized in 32 by 32 1-bit planes with corresponding bits in each plane connected to one PE. Data can also be passed between PEs along the grid. The cycle time of all PEs is 100 ns. This processor configuration is capable of performing ten billion binary operations per second; processing time increases proportionally with the precision of data items used. Two data mappings are particularly well suited to the DAP structure: a vector mode in which successive bits of a single word are mapped into a row of the array, and a matrix mode in which successive bits of a word are mapped into layers of the array memory vertically. Operations in both of these modes of operation are used in the system implementation presented in this paper.

The use of a massively parallel computer allows the application of techniques that would not be implemented on a serial computer due to their computational expense. Image processing and analysis can be done with fewer system cycles since many pixels of the image can be manipulated or analyzed at once. In the case of the DAP510c, up to 1024 pixels can be manipulated at the same time.

1.3 System Design

The present version of the system has been designed in a modular way so that different versions of each of the modules can be tested independently. This has allowed three methods of ridge direction feature extraction to be tested and has allowed K-L transforms and MLPs of several different sizes to be studied. This modular structure will allow the addition of a reject-testing method after the classification section; it has allowed networks to be developed that lump all arches into a single class at first and that separate A-T combinations with a second network. The modular structure separating image processing from classification has demonstrated its usefulness and should be retained. Substantial cost savings can result from this type of program design. The present system at NIST was constructed with about two staff-years of programing effort. This was possible by building on approximately 12 staff-years of effort spent on several character recognition system configurations [7]. This transfer of software expertise was possible because of the modular design of both systems.

1.4 Basic Models for Classification

In the past few years neural networks have been discussed as a possible method for constructing computer programs that can solve problems, like speech recognition and character recognition, where "human-like" response or artificial intelligence is needed. The most novel characteristics of neural networks are their ability to learn from examples, their ability to operate in parallel, and their ability

[1] DAP510c or equivalent commercial equipment may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

to perform well using data that are noisy or incomplete. In this paper, these characteristics of neural networks are illustrated using examples from the field of fingerprint classification. A functional input-output data flow of the system discussed in this paper is shown in table 1.

The neural approach to machine learning was originally devised by Rosenblat [8] by connecting together a layer of artificial neurons [9] on a perceptron network. The weaknesses which were present in this approach were analyzed by Minski and Papert [10] in 1962, and work on neurally based methods was abandoned by all but a small group of researchers for the next ten years. The advent of new methods for network construction and training in this ten year period led to rapid expansions in neural net research in the late 1980s.

Two types of learning, supervised learning and self-organization, are common in neural networks. The material presented in this paper does not cover the mathematical detail of these methods. A good source of general information on neural networks is Lippmann's review [11]. The primary research sources for neural networks are available in Anderson and Rosenfeld [12]. More detailed information on the supervised learning methods discussed here is given in [13]; self-organizing methods are discussed by Kohonen [14] and Grossberg [15].

The principal difference between neural network methods and rule-based methods is that the former attempts to simulate intelligent behavior by using machine learning and the latter uses logical symbol manipulation. Once the learning phase has been completed the network response is automatic and similar in character to reflex responses in living organisms. The processes where these methods have been most successful are in areas where human responses are automatic, such as touching one's nose or recognizing characters. Neural networks have been successful in engineering applications such as character recognition, speech recognition, and control systems for manufacturing, where information is incomplete and context dependent. In other areas, such as learning arithmetic [16], where automatic responses by humans are not expected, and precise answers are required, neural networks have had substantially less success.

It is important to understand that the accuracy of the final system produced will be strongly dependent on both the size and the quality of the training data. Many common test examples used to demonstrate the properties of pattern recognition system contain order of 10^2 examples. These examples show the basic characteristics of the system but provide only approximate idea of the syatem accuracy. The first version of the fingerprint system was built using 150 pairs of fingerprints. This sytem has accuracy of 94%. As the sample size was increased the accuracy initially dropped as more dificult cases were includes. As the test and training sample reached 1000 fingerprints the accuracy began to slowly improve. The porest accuracy achieved was with sample sizes near 10^3 and was 78%. The 2000 fingerprint sample used in this paper is well below the 10^5 fingerprint sample size which we have estimated is necessary to saturate the learning process.

2 Pattern Classification and Feature Extraction

The implementation of PCA discussed in this report combines several neural network methods to carry out the filtering and feature extraction. The filtering methods used are based on the ridge-valley method described in section 3 and on a method based on K-L transforms [2]. The basic classification process starts with an image of a fingerprint. In the self-organizing method, the ridge directions of the image are applied directly to the neural network and any filtering is learned as features are extracted [17]. In the supervised method, the features are extracted using the K-L transform, which is self-organizing, but classification is carried out using an MLP network.

An example of a two-class, two-feature problem using self-organization is shown in figure 2. The problem is to distinguish the arch class "A" from the whorl class "W." Two hundred samples of each fingerprint class are used in this example. A three-class, two-feature problem is shown in figure 3. In this example, the tented arch class "T" is added to the set of classes, and two hundred examples of it are added to the previous sample. Feature extraction in both examples is performed by correlating the input fingerprint image with the first two principal components, or eigenfunctions, using the K-L transform. One principal component is plotted on each axis of these figures. The self-organization method used calculates the distance from the mean of each class. If the distributions of the classes are

assumed to be normal, ellipses with major and minor axis lengths 2.17 times the standard deviation are expected to enclose 99% of each class. This method is a variation of the Probabilistic Neural Network [18]. Ellipses of these sizes are used in figures 2 and 3.

This is an extremely simple clustering method, but it illustrates the essential concepts of many self-organizing neural systems. In the A-W problem, the two classes are fairly well separated; only a small number of characters fall outside the 99% cluster boundaries. In the A-T-W problem, the "T"s overlap both the "A"s and part of the"W"s, but the overlap with the the "A"s is total. This is an illustration of the observation that "T"s look more like "A"s than "W"s. Both problems illustrate an important property of clustering methods: the ability to identify cases where the decision uncertainty is very high. Two types of uncertainty are shown. Uncertainty based on insufficient information is demonstrated by points outside any of the ellipses, the "W"s at the bottom of the 99% boundary. Uncertainty caused by confusion is demonstrated by points in two ellipses at once. In most self-organizing systems a point in this overlap region will be assigned to classes based on some correlation measure between the point and examples which have been previously stored. A point near the mean of the "A" class will have a high, but not certain, probability of being assigned that class. The self-organizing methods have no mechanism for using class information during training and so will cluster points entirely on the data stored in existing clusters.

The results of a linear classifier based on a single layer perceptron [8] for the same two problems are shown in figures 4 and 5. This method fits a linear decision surface between each of the classes; for a two-feature problem a single line is used. The method works well for the A-W problem but fails for the three class problem. The failure illustrates a fundamental problem with supervised systems. The error surfaces generated are "hard" surfaces. The method achieves an optimal fit as defined by the optimization method used on the learning data, but provides little information on the generalization capacity of the network. For the case shown, the "A" class is contained within the "T" class, and no line exists which will separate them using the first two K-L features. The goal of the supervised method is to draw the best possible boundary surface between classes. The surfaces used in this problem are restricted to lines by the simple linear optimization used. neural networks.

3 Image-Based Feature Extraction and Filtering

In this section two methods of feature extraction and image filtering are discussed. These methods are the ridge-valley filter and a Fourier-transform-based adaptive bandpass filter. The method used throughout the rest of this report is the ridge-valley filter. This choice was based on the filter cost for an acceptable level of feature extraction and image quality. The ridge-valley filter, implemented on a parallel computer, can process a 16 by 16 image tile in 256 μs. The Fourier transform filter, programed in assembly code on the same computer, can process a 32 by 32 image in 1 ms. The quality of image reconstruction available with these two methods is shown in figures 6 and 7. The original gray image is shown in figure 9.

The figures demonstrate that each of the methods remove some data and introduce some artifacts. The ridge-valley filtered image shown in figure 9 produces the image with the fewest artifacts. These artifacts are primarily white spaces in lines that change shape as a result of processing. The Fourier-transform-based filtered image shown in figure 7 introduces more artifacts. Most of these are associated with the 32 by 32 tile edges. A method for removing these artifacts in the FFT filter is discussed in section 3.2.

3.1 Ridge-Valley Features

This program is based on the "ridge-valley" fingerprint binarizer described in [19]. The binarizer works as follows. For each pixel C, slit sums $s_i, i = 1 \ldots 8$, are produced, where each s_i is the sum of the values of all the pixels labeled i in figure 8. The binarizer converts a gray-level fingerprint raster to a black and white version by a combination of "local thresholding" and "slit comparison" methods. The local thresholding formula sets the output pixel to white if the corresponding input pixel C exceeds

the average of the surrounding input pixels in the slits, that is, if

$$C > \frac{1}{32}\sum_{i=1}^{8} s_i. \tag{1}$$

The slit comparison formula sets the output pixel to white if the average of the maximum and minimum slit sums exceeds the average of all slit sums:

$$\frac{1}{2}(s_{max} + s_{min}) > \frac{1}{8}\sum_{i=1}^{8} s_i. \tag{2}$$

The motivation for equation 2 is as follows. A pixel in a light "valley" area of the print will have one of its slits lying along the valley for a high sum, and its other slits crossing ridges and valleys for roughly equal, lower sums; so the left side of equation 2 will exceed the right side. A similar reasoning applies to a pixel in a dark "ridge" area. In a combination of the two formulas, the output pixel is set to white if

$$4C + s_{max} + s_{min} > \frac{3}{8}\sum_{i=1}^{8} s_i. \tag{3}$$

The ridge-valley direction finder is a simple extension of the ridge-valley binarizer. Upon binarization, the ridge directions at the individual pixels are considered to be the directions of slits chosen in the natural way, that is, of the minimum-sum slits for pixels binarized to black and of the maximum-sum slits for pixels binarized to white. To produce a grid of directions spaced only every 16 pixels, these pixel directions are averaged over 16×16-pixel squares. Averaging has a smoothing effect and produces a finer direction quantization.

The ridge angle θ at a location is defined to be 0° if the ridges are horizontal, increasing towards 180° as the ridges rotate counterclockwise, and reverting to 0° when the ridges again become horizontal: $0° \leq \theta < 180°$. When pixel directions are averaged, the quantities averaged are actually not the pixel ridge angles θ, but rather the pixel "direction vectors" $(\cos 2\theta, \sin 2\theta)$. Averaging the angles can produce absurd results: the average of a 1° direction and a 179° direction, each nearly horizontal, produces a vertical 90° direction. Averaging the cosines and sines of the angles also fails: for the same 1° and 179° directions, the (cosine, sine) vectors are (0.999848, 0.017452) and (-0.999848, 0.017452), whose average is the vertical (0, 0.017452). However, good results are obtained by doubling the angles and then taking cosines and sines. Using this method, 1° and 179° ridges become (0.999391, 0.034900) and (0.999391, -0.034900), which average to the horizontal (0.999391, 0).

The direction finder produces a grid of averages of pixel direction vectors. Since the pixel direction vectors have length 1, each average vector's length is at most 1. If a region of the fingerprint has a poorly defined ridge direction, say because of overinking, the directions at its several pixels tend to cancel each other out and produce a short average vector.

An earlier version of the direction finder produced a grid of directions spaced 16 pixels apart horizontally and vertically, for a total of 840 (28 by 30) vectors. The current version produces better classification results by using the same number of vectors, but arranged in a fixed unequally-spaced pattern which concentrates the vectors in certain areas at the expense of less important areas. Each 32×32-pixel tile of the raster gets either 1, 4, or 16 direction vectors. First, a grid is produced with the vectors spaced every 8 pixels (but still using 16×16-pixel averaging windows); this grid has 16 vectors per tile. Grids with 4 vectors/tile and 1 vector/tile are produced from this original grid by two averaging steps. Then, some tiles receive their vectors from the coarse grid, some from the medium grid, and some from the fine grid, according to a pattern produced as follows. Let the number of tiles that receive 1, 4, and 16 vectors be n_1, n_4, and n_{16}. There are $15 \times 16 = 240$ tiles, so $n_1 + n_4 + n_{16} = 240$. The total number of vectors is fixed at 840 for comparability with the earlier version, so $n_1 + 4n_4 + 16n_{16} = 840$. Using these two equations in three variables, integer values of n_{16} with $0 \leq n_{16} \leq 40$ produce n_1 and n_4 values that are nonnegative integers. Meaningful values for the three variables were produced by simply picking n_{16} values and solving for the other two variables, since there is not a unique meaningful solution.

The equations determine the numbers of tiles that are to be allotted 1, 4, and 16 vectors (given n_{16}), but do not indicate which tiles should get which numbers of vectors. This was settled by using cores and deltas. Cores and deltas have ridge-directions that change unusually rapidly, so it seemed reasonable to concentrate the vectors in tiles likely to contain cores and deltas. The approximate positions of the cores and deltas of some fingerprints had already been manually marked for another experiment; the R92-P registration program was run on these fingerprints and each core or delta location was translated by the dx and dy by which R92-P would have translated the fingerprint to register it. A histogram was made of the resulting adjusted core and delta locations and the tiles were sorted accordingly. The n_1 lowest-scoring tiles (fewest cores or deltas) each received 1 vector, the next n_4 tiles, 4 vectors, and the n_{16} highest-scoring tiles, 16 vectors.

Values of 5, 10, and 15 were tried for n_{16}; in each case, a K-L transform was produced and the resulting features were used with the conjugate-gradient MLP classifier, using a few different numbers of input and hidden nodes. Using $n_{16} = 10$, 80 input nodes (K-L features), 96 hidden nodes, and of course 5 output nodes for the 5 classes, the result was 86% correct classifications when no examples were rejected. These results were obtained using an fft-based preprocessing filter described in section 3.3 followed by ridge direction extraction for the R92-P program that registers the fingerprints before extracting the unequally-spaced grid. When the same filter and registration were used but followed by the equally-spaced grid, the best result obtained across several net architectures was 83% correct.

Equally-spaced grids of directions for five fingerprints are shown in figures 9-13. The unequally-spaced grid for fingerprint f0009_08 is shown in figure 14.

As implemented on the DAP, the ridge-valley direction finder for an equally-spaced pattern of 840 ridge directions takes about 0.22 seconds, and the finder for an unequally-spaced pattern of 840 directions takes a moderate amount longer.

3.2 FFT Fingerprint Preprocessing Filter

This filter is used to improve the quality of the fingerprint image before extracting the ridge directions. Its algorithm is basically the same as the one described in [20].

The filter processes the image in tiles of 32×32 pixels, starting in the upper left hand corner of the image. After extracting a tile the filter shifts right 24 pixels to obtain the next 32×32 tile, resulting in the first 8 columns of data in the new tile being common with the last 8 columns of the previous 32×32 tile. After reaching the right side of the image the filter shifts down 24 pixels, resulting in 8 rows of common data between vertically adjacent tiles, and restarts at the left side of the image working toward the right. Processing continues in this manner until reaching the bottom right corner of the image. The common data between the horizontally and vertically adjacent tiles helps reduce the artifacts created by processing the image in tiles.

Each tile is thought of as a matrix A of real numbers. To filter a tile, the first step is to compute the discrete two-dimensional forward Fourier transform, defined as follows (with B set to zeros):

$$X_{jk} + iY_{jk} = \sum_{m=1}^{32} \sum_{n=1}^{32} (A_{mn} + iB_{mn}) \exp\left(2\pi i \left(\frac{(j-1)(m-1)}{32} + \frac{(k-1)(n-1)}{32}\right)\right)$$

(A "fast Fourier transform" (fft) routine is used, rather than using the above formula directly.) Some of the fft elements, corresponding to very low and very high spatial frequencies, are set to zero, thereby filtering out these extreme frequencies. The "power spectrum" P of the fft is computed:

$$P_{jk} = X_{jk}^2 + Y_{jk}^2$$

The elements of P are then raised to a power α and multiplied by the fft elements $X + iY$ to produce new elements $U + iV$:

$$Q_{jk} = P_{jk}^{\alpha}$$

$$U_{jk} = Q_{jk} X_{jk}$$

$$V_{jk} = Q_{jk} Y_{jk}$$

Finally, the inverse Fourier transform of $U + iV$ is computed, and its real part becomes the filtered tile, which is used to reconstruct the "filtered" image. In reconstructing the image, the filter accounts for the 8 columns/rows of common data between adjacent tiles by only keeping the center 24 $\times$ 24 pixels and discarding the outer 4 rows/columns from each (32 $\times$ 32) filtered tile. The multiplication of the fft elements by a power of the power spectrum has the effect of causing those frequencies of a tile that were originally dominant to become even more dominant. Presumably, the dominant frequencies of a fingerprint tile are those corresponding to the ridges; so, this filter tends to increase the ratio of ridge information to non-ridge noise. It adapts to variations of the ridge-frequency from one tile of a fingerprint to another.

3.3 FFT Ridge-Direction Finder

This program is used to compute, for each tile of a fingerprint image, the local direction of the ridges. As in the preprocessing filter of the preceding section, the fft $X + iY$ of the tile, and its power spectrum P, are computed. We observed that in a conventional image such as figure 15, the power spectrum of a typical tile peaks at the lowest frequencies and falls off as $1/f$, as shown in figure 16. In a fingerprint, however, the power spectrum of a tile almost always has two distinct peaks on either side of the center DC value, as shown in figure 17. A line connecting either spot to the center of P is perpendicular to the ridges. Therefore, each element of P corresponds to a ridge angle. It seems reasonable to compute the ridge angle as a weighted average of the angles corresponding to all the elements, with the P values at the elements used as weights. However, for the same reasons as mentioned above in the discussion of the ridge-valley direction finder, it is better to compute an average of "direction vectors" rather than of angles. The result is an average ridge direction vector for the tile.

A couple of tricks were used to increase speed of this program. First, a way was found to compute two real-input fft's using only one call of the fft routine, by exploiting fft symmetries. Second, it was possible to combine two input tiles by multiplying only one of them by a "checkerboard" pattern of 1's and -1's and then adding them, taking just one fft, and then separating the resulting power spectrum into parts corresponding to the two tiles. The checkerboarded tile's low-frequency elements are near the middle of the fft and the other tile's low-frequency elements are near the corners; since fingerprint tiles are very weak in high frequencies, the far-from-zero parts of the two patterns do not overlap and they can therefore be snipped apart and recovered with little error. Combining these two shortcuts allowed us to process four tiles with only one call of the fft routine.

As implemented on the DAP, this method took about 0.48 seconds to assign 256 ridge directions to a fingerprint. The ridge-valley direction finder (in the version that produces an equally-spaced grid) takes only 0.22 seconds to produce a grid of 840 directions; if the fft direction finder had to produce 840 directions, it would take about 1.58 seconds. The ridge-valley direction finder also produced results that appeared to be of higher quality. Because the ridge-valley direction finder is superior to the fft direction finder both in speed and quality, we have discontinued work on the fft direction finder.

4 Classification of Fingerprints

4.1 The Layered Perceptron Networks

The two layer perceptron nonlinearly classifies K-L feature vectors. With the evolved K-L feature extraction, the network may be regarded as the three layer fingerprint classifier of figure 18. The first set of weights Ψ_0 is the *pre-trained* incomplete eigenvector basis set. The latter perceptron weight layers, also fully interconnected, are trained using the conjugate gradient algorithm [6].

All the Karhunen Loève transform vectors are propagated through the network together and the weights are updated. This is *batch mode* training. The use of different subsets of the training patterns to calculate each weight update is known as *on-line* training. It is not used in this investigation. Formally, the forward propagation is represented as:

$$\mathbf{V_1} = \mathbf{\Psi_0^T V_0} \Longrightarrow \mathbf{V_2} = \mathbf{f}\left(\mathbf{\Psi_1^T V_1}\right) \Longrightarrow \mathbf{V_3} = \mathbf{f}\left(\mathbf{\Psi_2^T V_2}\right) \tag{4}$$

where the network nonlinearity is introduced by squashing all activations with the usual sigmoid function $f(x) = (1 + \epsilon^{-x})^{-1}$.

4.2 Classification of Fingerprints

The linear superposition of a complete set of orthogonal basis functions will exactly reproduce an arbitrary eigen fingerprint. However, the whole motivation for using the KLT is to reduce the dimensionality of the feature space by adopting an incomplete basis, i.e., the leading principal components. Only images that resemble the original training ridge-valley directions are adequately representable by the reduced basis. It is important that the eigenvectors are obtained from a statistically large sample since adequate sampling of the feature set is required for this reduction to be useful.

4.3 Conjugate Training Algorithm for Feed Forward Neural Networks

Backpropagation [13] is the most common method for training MLP networks. Essentially, it implements a first order minimization of some error objective. The algorithm has the disadvantages that convergence is slow [21] and that there are, in the usual implementation [13], two adjustable parameters, η and α, that have to be manually optimized for the particular problem.

Conjugate gradient methods have been used for many years [22] for minimizing functions, and have recently [23] been discovered by the neural network community. The usual methods require an expensive line search or its equivalent. Møller [24] has introduced a scaled conjugate gradient method that instead of a line search uses an estimate of the second derivative along the search direction to find an approximation to the minimum error along the search direction. In both backpropagation and scaled conjugate gradient, the most time-consuming part of the calculation is done by the forward error and gradient calculation. In backpropagation this is done once per iteration. Although the scaled conjugate gradient method does this calculation twice per iteration, the factor of two overhead is algorithmically negligible since convergence is an order of magnitude faster for OCR and fingerprint classification [24] [6].

4.4 Fingerprints Used for Training and Testing

All fingerprints used in this study were from NIST Special Database 4 [25]. This database consists of 4000 8-bit raster images made from 2000 different fingers, with each finger represented as two different impressions ("rollings") that were made on different occasions. The images were chosen in such a way that each of the five classes is equally represented (400 pairs of rollings.) Each image in the database is labeled with its correct class as determined by an expert. (Of course, the two rollings of any finger are of the same class.) Supervised training of the MLP was done using features derived from first-rolling fingerprints, and testing was done using features from the second-rollings.

4.5 Karhunen-Loève Transform

The Karhunen-Loève (K-L) Transform [26] was used to reduce the 1680 ridge-direction features of a fingerprint to a much smaller number of features, such as 64 or 96. This greatly reduces the number of weights in the MLP, and thereby reduces the amount of time required for training.

4.6 Testing Results

4.6.1 *A Priori* Probabilities and Realistic Scoring

The obvious way to produce a test score is to simply use the percentage of the test set that was classified correctly. However, this score is useful as a predictor of subsequent performance on naturally occuring fingerprints only if the test set is statistically the same as natural fingerprints. This is not the case for Special Database 4. It has equal numbers of prints from each class, but the *a priori* probabilities of natural fingerprints (that is, the proportions of them belonging to each class) are far from equal. In

fact, the *a priori* probabilities, as we have calculated them from a classification summary of more than 222 million prints, are approximately .037, .338, .317, .029, and .279 for the classes A, L, R, T, and W.

The following calculations were used to produce a realistic test score using this unnatural test set. Each of the following quantities is defined for classes $i = 1, \ldots, 5$:

$$\begin{aligned} p_i &= \textit{a priori} \text{ prob. of class } i \\ c_i &= \text{number of the 400 test examples of class } i \text{ that were correctly classified} \\ P(c|i) &= \text{conditional prob. of correct classification, given that actual class is } i \\ P(c) &= \text{prob. of correct classification} \end{aligned}$$

By definition,

$$P(c) = \sum_{i=1}^{5} p_i P(c|i).$$

And clearly, $c_i/400$ is a reasonable estimate of $P(c|i)$. Therefore, we can estimate that:

$$P(c) \approx \sum_{i=1}^{5} p_i c_i/400.$$

To test the MLP's performance and find the best architecture, the numbers of input and hidden nodes were each varied from 32 through 96 in steps of 16. Each of the resulting 25 network architectures was trained with five different seeds for random generation of initial weights, to assure that at least one of its training runs produced a representative result for that architecture. Table 2 shows the test scores; the 64-96-5 architecture produced the best score.

4.6.2 Rejection

The classifier can be set up to reject some of the prints it is presented with, so as to reduce the proportion of the accepted prints that it misclassifies. Our rejection mechanism is a simple algorithm: it rejects a print if the highest output "activation" of the MLP minus the second-highest activation is below a threshold.

To implement various levels of rejection, the threshold is set to different levels. The overall performance of the classifier across many levels of rejection can be summarized in a "correct vs. rejection" curve, which shows the "proportion of accepted prints correctly classified" vs. the "proportion of prints rejected". Actually, we have used calculations similar to those of the preceding section to produce a graph of "estimated conditional probability of correct classification given acceptance" vs. "estimated probability of rejection". This curve reflects expected performance on natural fingerprints, even though it is produced using the unnatural test set. The curve for the winning 64-96-5 architecture is shown in 19.

4.7 *A Priori* Probabilities and Better Training

At first, we trained on all 2000 first-rolling prints in the database. Then, it occured to us that better results might be obtained by causing the training set to have a natural distribution. The new training set was made by using all 400 prints of class L (the most common class) and discarding some of the prints from each of the other classes, so that the proportion of the resulting set belonging to each class was approximately equal to the *a priori* probability of that class. This modified training set did in fact produce better results. All results reported here were made using this new training set.

5 Conclusions

5.1 Testing Requirements

All work performed in this study has used NIST Special Database 4 [25]. The sample size available from this database should be sufficient to test PCA accuracy to about the 2-3% error level. We estimate that further testing to reduce error to the 0.3% level will require sample sizes from 25 to 81 times as large as the one presently available.

5.2 Accuracy

Accuracy is the most difficult part of the PCA problem. The best accuracy achieved to date on a balanced sample, with equal numbers of A-T-L-R-W patterns, is 90.2% with 10% rejects. When the a prori probabilities of the different classes are taken into account this is increased to 95.4% with 10% rejects. To put this result into perspective, 89% is approximately the level of accuracy achieved with handprinted digits using Gabor features [27] in early 1991. Results achieved on the handprint digit problem, by expanding the training and testing sets and by using better segmentation and feature extraction, have allowed accuracy on character recognition to improve to 98.96% with 10% rejects. This suggests that a PCA system can be built which will achieve 99% accuracy.

5.3 Speed

The speed achieved in this study, 2.65 seconds per fingerprint, demonstrates that existing parallel computers are fast enough to process the FBI's current workload with a small number of systems. This speed is also essential for large scale testing of potential recognition methods. If feature extraction using ridge directions or some other method takes 1000 seconds per fingerprint instead of 2.65 second per fingerprint, the training time for the existing database goes from 50 minutes to over 550 hours, or 23 days. This demonstrates that the extensive training and testing performed on 2000 fingerprint samples required for this study is practical only if speeds near the FBI current operative requirement are achieved.

Acknowledgement

The authors would like to acknowledge Jim Blue, Jane Walters, Jon Geist, Mike Gilcrest, and Fred Preston for assistance in improving the presentation and clarity of this paper.

References

[1] *The Science of Fingerprints.* U. S. Department of Justice, Washington, DC, 1984.

[2] P. J. Grother. Karhunen Loève feature extraction for neural handwritten character recognition. In *Proceedings: Applications of Artificial Neural Networks III.* Orlando, SPIE, April 1992.

[3] R. Linsker. Self-organization in a perceptual network. *Computer*, 21:105–117, 1988.

[4] M. V. Wickerhauser. Fast approximate factor analysis. In *Proceedings October 1991.* SPIE, Washington University in St. Louis, Department of Mathematics, 1991.

[5] T. P. Vogl, K. L. Blackwell, S. D. Hyman, G. S. Barbour, and D. L. Alkon. Classification of Japanese Kanji using principal component analysis as a preprocessor to an artificial neural etwork. In *International Joint Conference on Neural Networks*, volume 1, pages 233–238. IEEE and International Neural Network Society, 7 1991.

[6] J. L. Blue and P. J. Grother. Training Feed Forward Networks Using Conjugate Gradients. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 179–190, San Jose California, February 1992. SPIE.

[7] M. D. Garris, C. L. Wilson, J. L. Blue, G. T. Candela, P. Grother, S. Janet, and R. A. Wilkinson. Massivelly parallel implementation of character recognition systems. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 269–280, San Jose California, February 1992. SPIE.

[8] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[9] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics*, 9:115–133, 1943.

[10] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[11] R. P. Lippman. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

[12] J. A. Anderson and E. Rosenfeld. *Neurocomputing*. MIT Press, Cambridge, MA, 1989.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, et al., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[14] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, second edition, 1988.

[15] S. Grossberg. On learning and energy-entropy dependence in recurrent and nonrecurrent signed networks. *J. Statistical Physics*, 1:319–350, 1969.

[16] J. A. Anderson, M. L. Rossen, S. R. Viscuso, and M. E. Sereno. Experiments with the representation in neural networks: Object, motion, speech, and arithmetic. In H. Fanken and M. Stadler, editors, *Synergetics of Cognition*, pages 54–69. Springer-Verlag, Berlin, 1990.

[17] C. L. Wilson. FAUST: a vision based neural network multi-map pattern recognition architecture. In *Proceedings: Applications of Artificial Neural Networks III*. Orlando, SPIE, April 1992.

[18] Donald F. Specht. Probalistic neural networks. *Neural Networks*, 3(1):109–118, 1990.

[19] R. M. Stock and C. W. Swonger. Development and evaluation of a reader of fingerprint minutiae. *Cornell Aeronautical Laboratory*, Technical Report CAL No. XM-2478-X-1:13–17, 1969.

[20] Automated classification system reader project (ACS). Technical report, DeLaRue Printrak Inc., February 1985.

[21] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[22] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[23] E. M. Johansson, F. U. Dowla, and D. M. Goodman. Backpropagation learning for multi-layer feed-forward neural networks using the conjugate gradient method. *IEEE Transactions on Neural Networks*, 1991.

[24] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 1991. to be published.

[25] C. I. Watson and C. L. Wilson. Fingerprint database. *National Institute of Standards and Technology*, Special Database 4, **FPDB**, April 18, 1992.

[26] Anil K. Jain. *Fundamentals of Digital Image Processing*, chapter 5.11, pages 163–174. Prentice Hall Inc., prentice hall international edition, 1989.

[27] M. D. Garris, R. A. Wilkinson, and C. L. Wilson. Methods for enhancing neural network handwritten character recognition. In *International Joint Conference on Neural Networks*, volume 1, pages 695–700. IEEE and International Neural Network Society, 7 1991.

System Module	Input	Output	Time (sec.)
FFT Filter	8-bit 512×512 Image	filtered Image	1.73
Ridge Valley Features	8-bit 512×512 Image	840 directions (unregegestered)	0.3
R92 Resistration	840 directions	offset Image	0.1
Registered Ridge Valley Features	x, y translated Image	840 directions (registered)	0.216
K-L Transform	840 directions	64 features	0.02
MLP Classifier	64 features	5 classes	0.001

Table 1: Input-Output Data flow for a fingerprint classification system based on supervised learning.

inputs	*hiddens*				
	32	48	64	80	96
32	90.29	89.82	90.96	90.24	90.99
48	90.54	90.24	91.34	91.12	90.92
64	90.66	90.48	91.20	91.19	91.76
80	90.60	91.34	90.60	90.27	91.18
96	91.07	90.65	90.40	91.28	90.37

Table 2: Testing Percentages Correct for Various Architectures

Processor Elements Array and the Array Memory

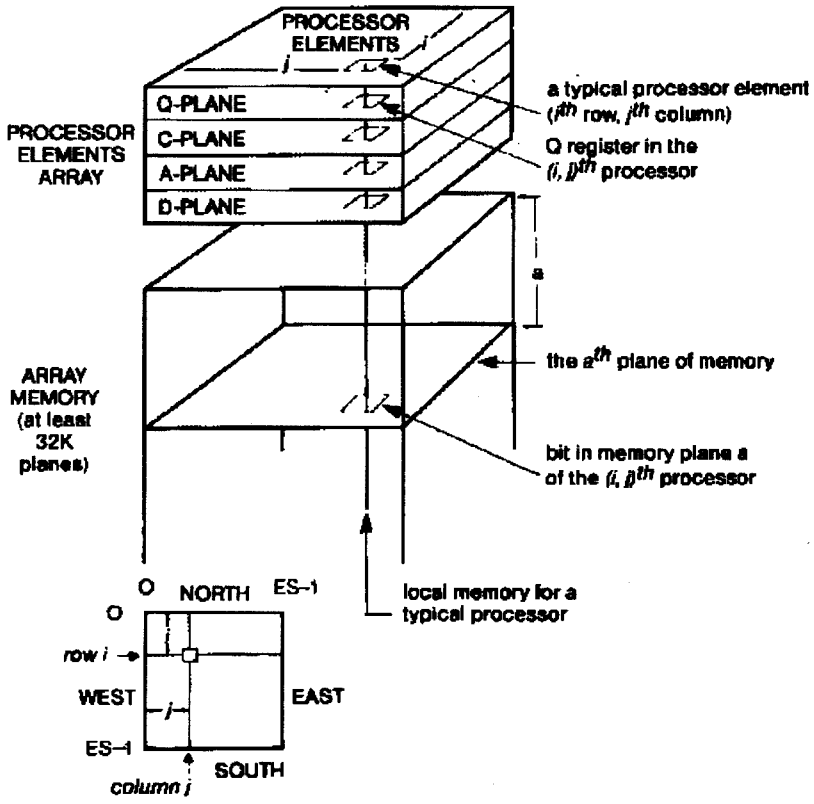

Figure 1: Array processor architecture for a massively parallel computer.

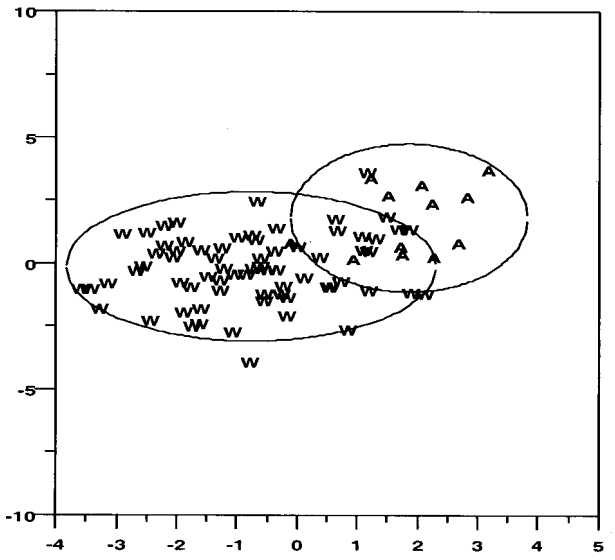

Figure 2: Self-organized clusters for the A-W classification problem.

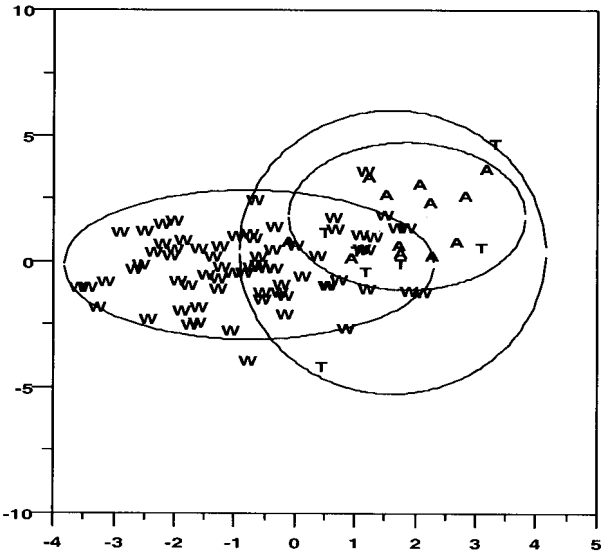

Figure 3: Self-organized clusters for the A-T-W classification problem.

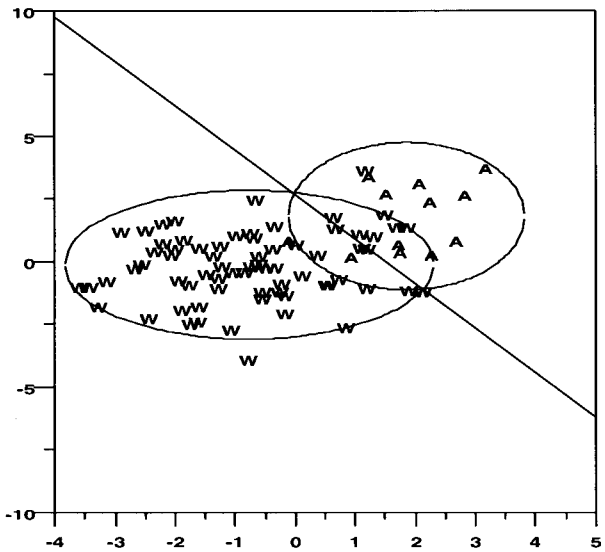

Figure 4: Supervised linear model for the A-W classification problem superimposed on the previous self-organizing clusters.

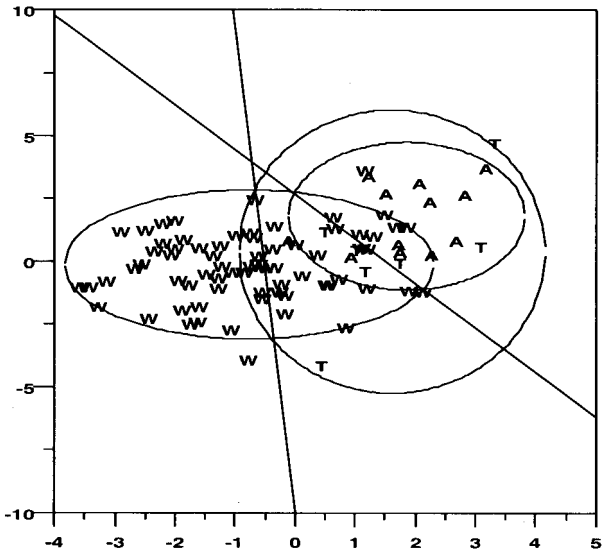

Figure 5: Supervised linear model for the A-T-W classification problem superimposed on the previous self-organizing clusters.

Figure 6: Example of ridge-valley filtering of a fingerprint image.

Figure 7: Example of Fourier transform based filtering of a fingerprint image.

7		8		1		2		3
6		7	8	1	2	3		4
		6				4		
5		5		C		5		5
		4				6		
4		3	2	1	8	7		6
3		2		1		8		7

Figure 8: Ridge-valley pixel arrangement.

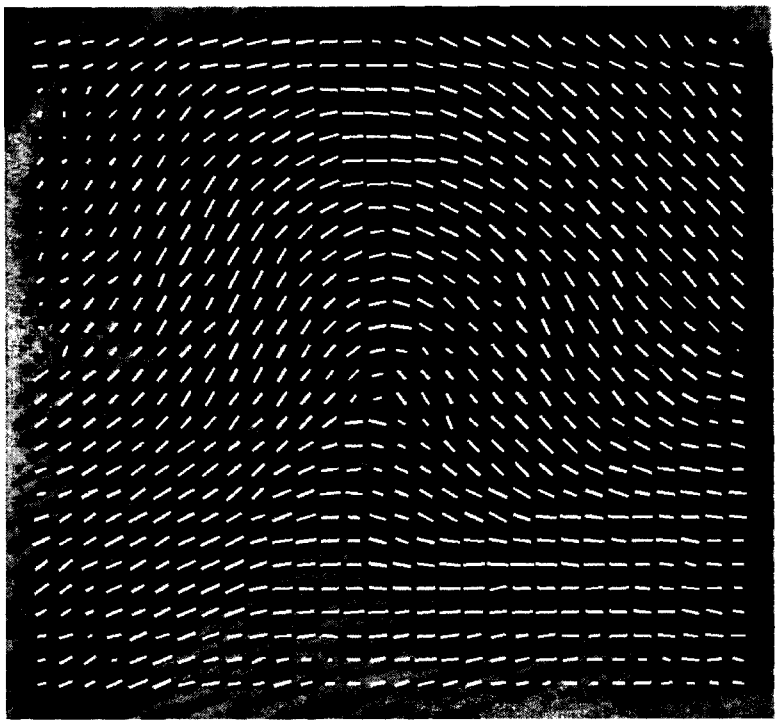

Figure 9: Examples of an arch fingerprint image with ridge directions from the ridge-valley algorithm.

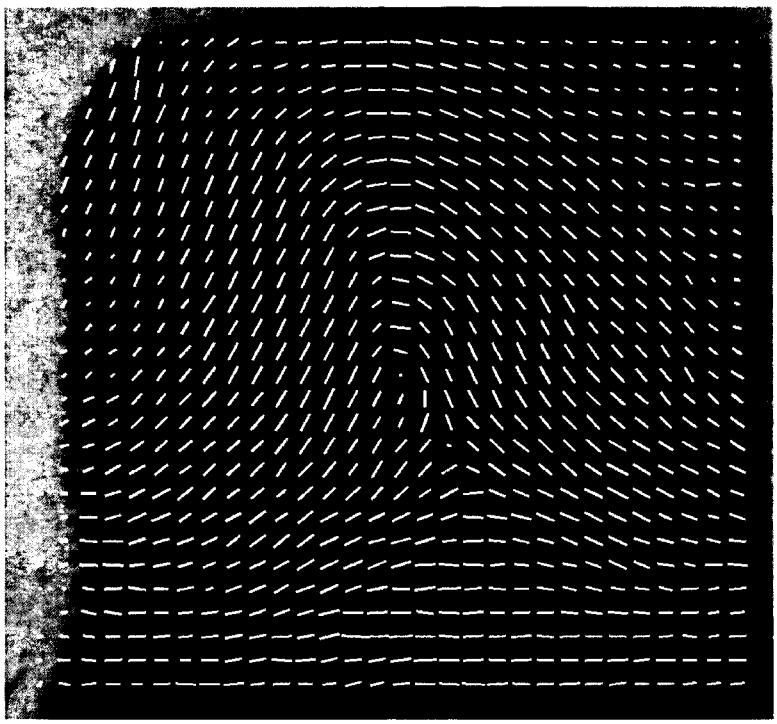

Figure 10: Examples of a left loop fingerprint image with ridge direction from the ridge-valley algorithm.

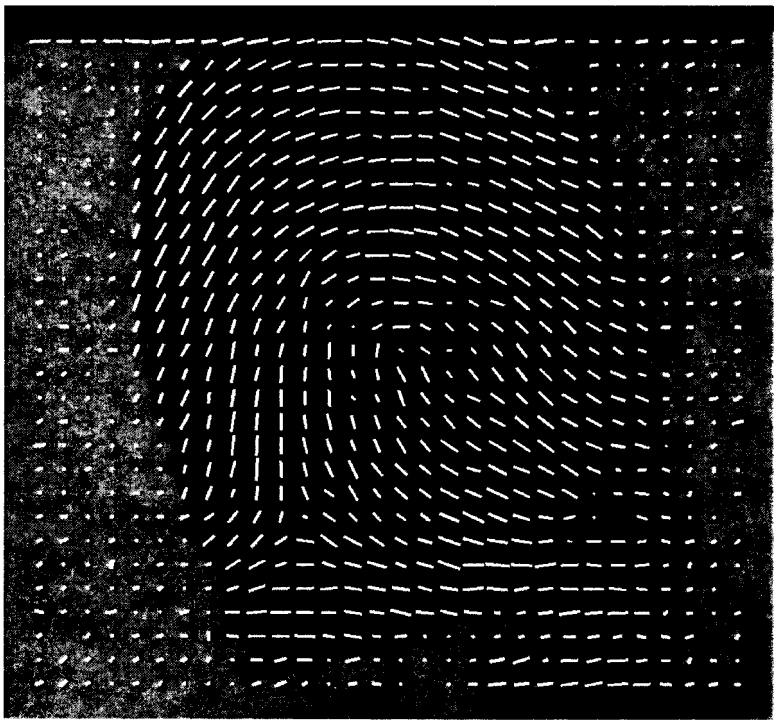

Figure 11: Examples of a right loop fingerprint image with ridge direction from the ridge-valley algorithm.

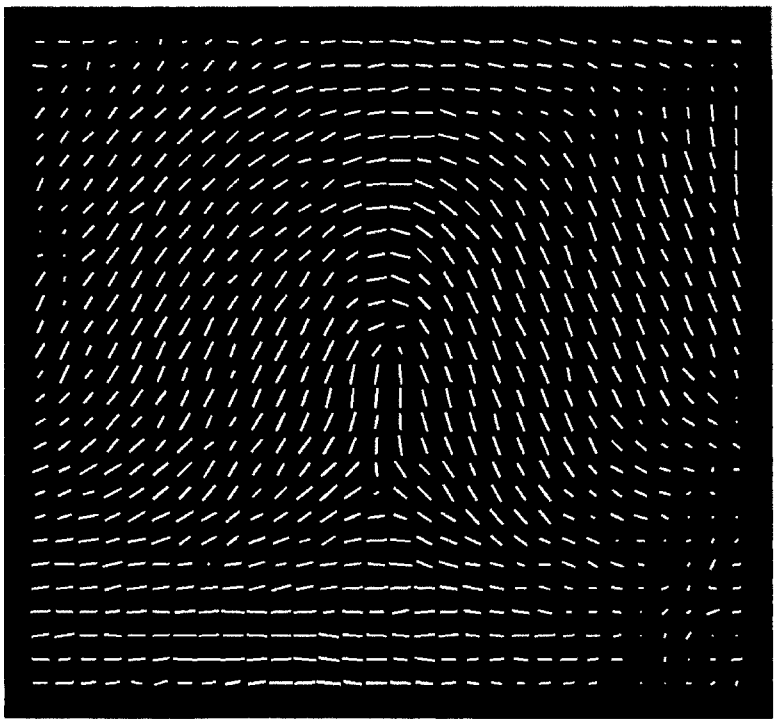

Figure 12: Examples of a tented arch fingerprint image which is cross referenced to a right loop with ridge direction from the ridge-valley algorithm.

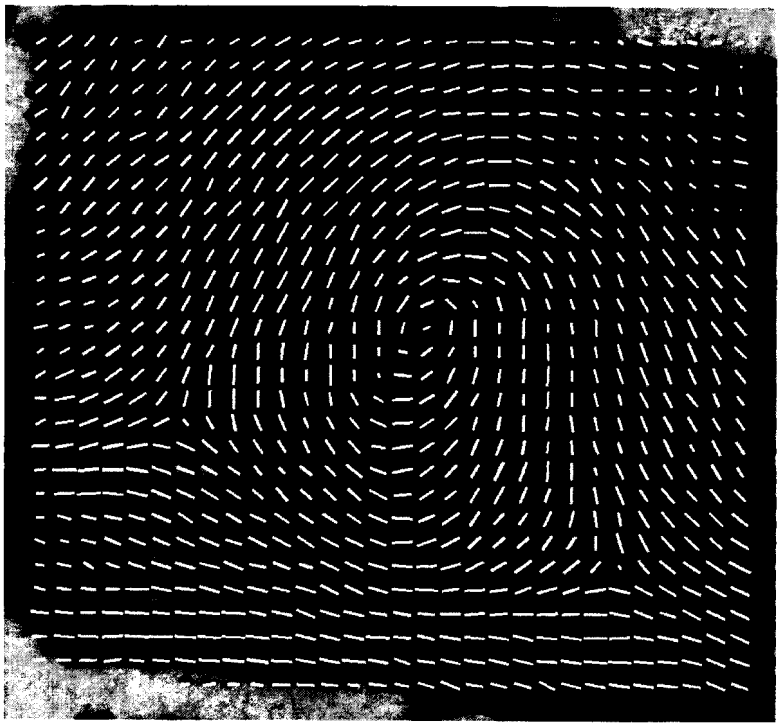

Figure 13: Examples of a whorl fingerprint image with ridge direction from the ridge-valley algorithm.

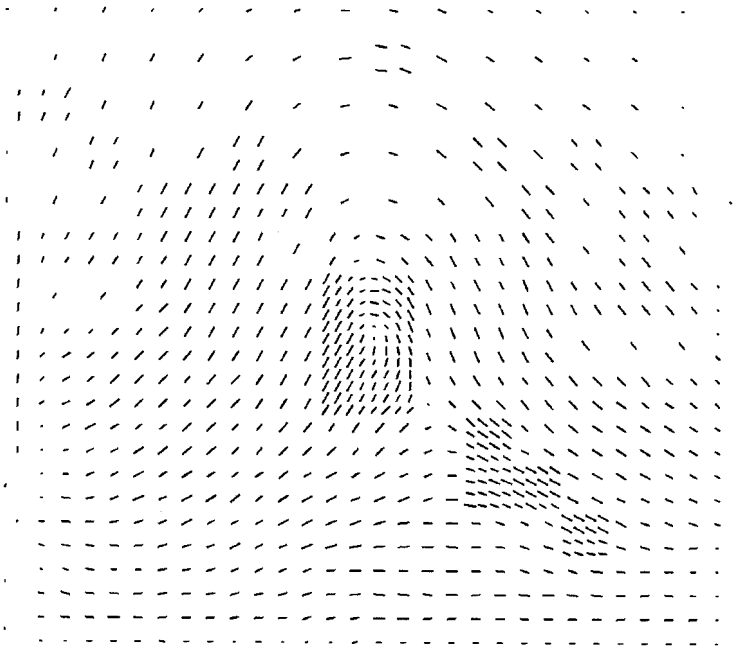

Figure 14: Examples of a non-uniform ridge-valley mesh.

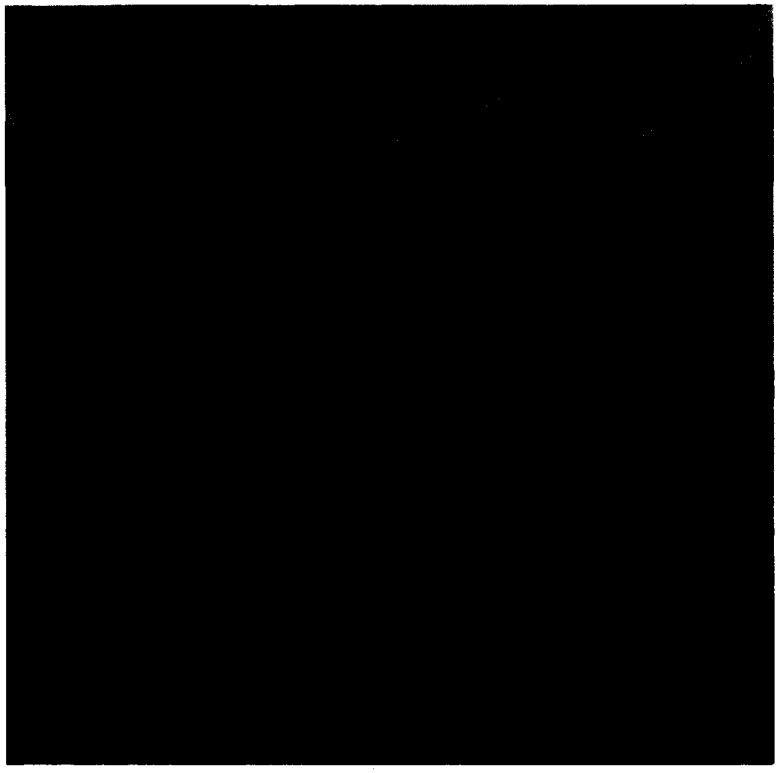

Figure 15: Example image of a space walk.

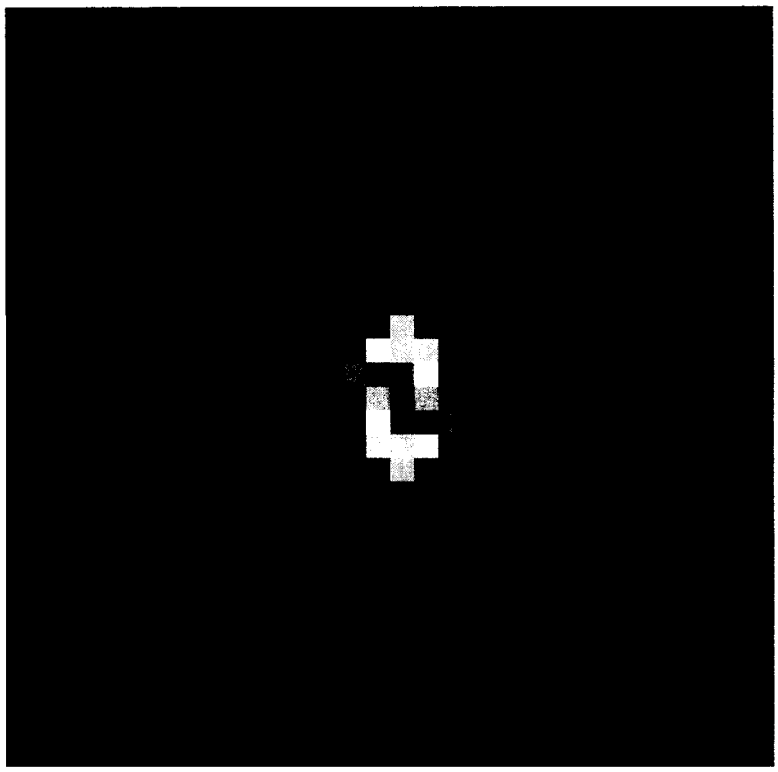

Figure 16: Example of Fourier transform power spectrum of a 32×32 tile of the space walk image. DC is at the center of the image and high frequencies are at the corners.

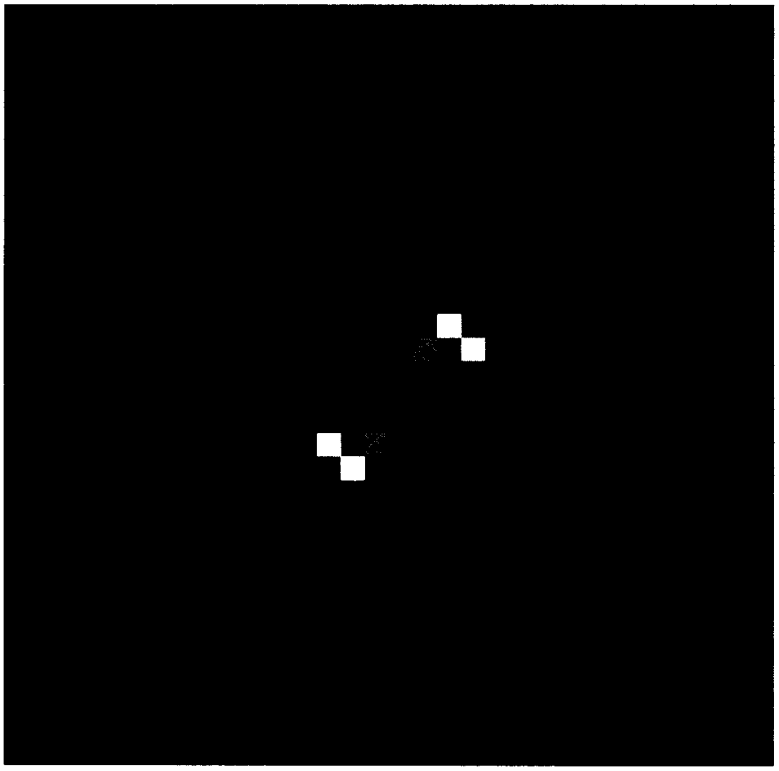

Figure 17: Example of Fourier transform power spectrum of a 32×32 tile of a fingerprint image. DC is at the center of the image and high frequencies are at the corners.

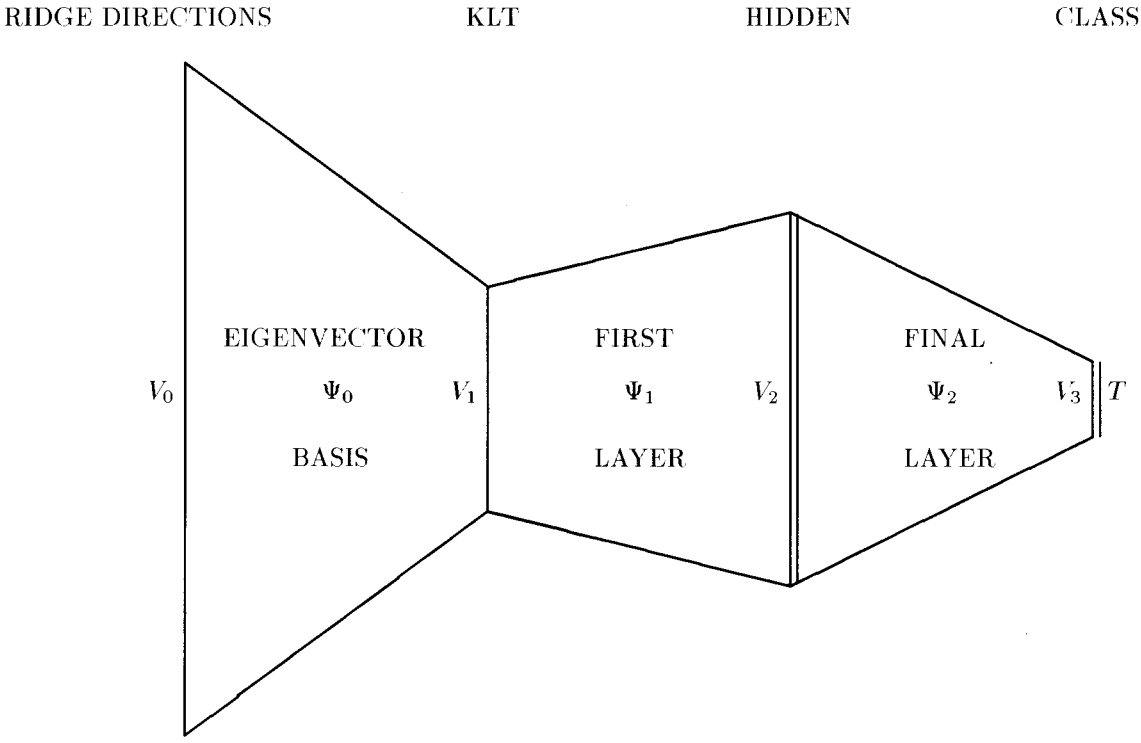

Figure 18: Classification Architecture. All weight layers are fully connected. The eigenvectors are obtained a priori to the training of the subsequent layers.

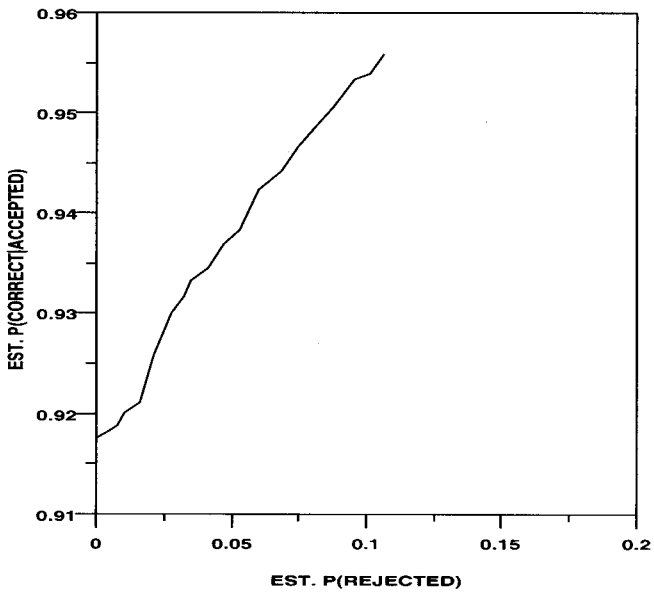

Figure 19: Reject versus accuracy curve for 64-96-5 network