Face Recognition: A Hybrid Neural Network Approach

Steve Lawrence[1,2]*, C. Lee Giles[1]†, Ah Chung Tsoi[2], Andrew D. Back[2]
{lawrence,act,back}@elec.uq.edu.au, giles@research.nj.nec.com

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[2] Electrical and Computer Engineering, University of Queensland, St. Lucia, Australia

Technical Report
UMIACS-TR-96-16 and CS-TR-3608
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract

Faces represent complex, multidimensional, meaningful visual stimuli and developing a computational model for face recognition is difficult [42]. We present a hybrid neural network solution which compares favorably with other methods. The system combines local image sampling, a self-organizing map neural network, and a convolutional neural network. The self-organizing map provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, thereby providing dimensionality reduction and invariance to minor changes in the image sample, and the convolutional neural network provides for partial invariance to translation, rotation, scale, and deformation. The convolutional network extracts successively larger features in a hierarchical set of layers. We present results using the Karhunen-Loève transform in place of the self-organizing map, and a multi-layer perceptron in place of the convolutional network. The Karhunen-Loève transform performs almost as well (5.3% error versus 3.8%). The multi-layer perceptron performs very poorly (40% error versus 3.8%). The method is capable of rapid classification, requires only fast, approximate normalization and preprocessing, and consistently exhibits better classification performance than the eigenfaces approach [42] on the database considered as the number of images per person in the training database is varied from 1 to 5. With 5 images per person the proposed method and eigenfaces result in 3.8% and 10.5% error respectively. The recognizer provides a measure of confidence in its output and classification error approaches zero when rejecting as few as 10% of the examples. We use a database of 400 images of 40 individuals which contains quite a high degree of variability in expression, pose, and facial details. We analyze computational complexity and discuss how new classes could be added to the trained recognizer.

Keywords: Convolutional Networks, Hybrid Systems, Face Recognition, Self-Organizing Map

* http://www.neci.nj.nec.com/homepages/lawrence, http://www.elec.uq.edu.au/~lawrence
† Also with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

1 Introduction

The requirement for reliable personal identification in computerized access control has resulted in an increased interest in biometrics[1]. Biometrics being investigated include fingerprints [4], speech [7], signature dynamics [36], and face recognition [8]. Sales of identity verification products exceed $100 million [29]. Face recognition has the benefit of being a passive, non-intrusive system for verifying personal identity. The techniques used in the best face recognition systems may depend on the application of the system. We can identify at least two broad categories of face recognition systems:

1. We want to find a person within a large database of faces (eg. in a police database). These systems typically return a list of the most likely people in the database [34]. Often only one image is available per person. It is usually not necessary for recognition to be done in real-time.

2. We want to identify particular people in real-time (eg. in a security monitoring system, location tracking system, etc.), or we want to allow access to a group of people and deny access to all others (eg. access to a building, computer, etc.) [8]. Multiple images per person are often available for training and real-time recognition is required.

In this paper, we are primarily interested in the second case[2]. We are interested in recognition with varying facial detail, expression, pose, etc. We do not consider invariance to high degrees of rotation or scaling - we assume that a minimal preprocessing stage is available if required. We are interested in rapid classification and hence we do not assume that time is available for extensive preprocessing and normalization. Good algorithms for locating faces in images can be found in [42, 40, 37].

The remainder of this paper is organized as follows. The data we used is presented in section 2 and related work with this and other databases is discussed in section 3. The components and details of our system are described in sections 4 and 5 respectively. We present and discuss our results in sections 6 and 7. Computational complexity is considered in section 8 and we draw conclusions in section 10.

2 Data

We have used the ORL database which contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK[3]. There are 10 different images of 40 distinct subjects. For some of the subjects, the images were taken at different times. There are variations in facial expression (open/closed eyes, smiling/non-smiling), and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an up-right, frontal position, with tolerance for some tilting and rotation of up to about 20 degrees. There is some variation in scale of up to about 10%. Thumbnails of all of the images are shown in figure 1 and a larger set of images for one subject is shown in figure 2. The images are greyscale with a resolution of 92x112.

[1]Physiological or behavioral characteristics which uniquely identify us.

[2]However, we have not performed any experiments where we have required the system to reject people that are not in a select group (important, for example, when allowing access to a building).

[3]The ORL database is available free of charge, see `http://www.cam-orl.co.uk/facedatabase.html`.

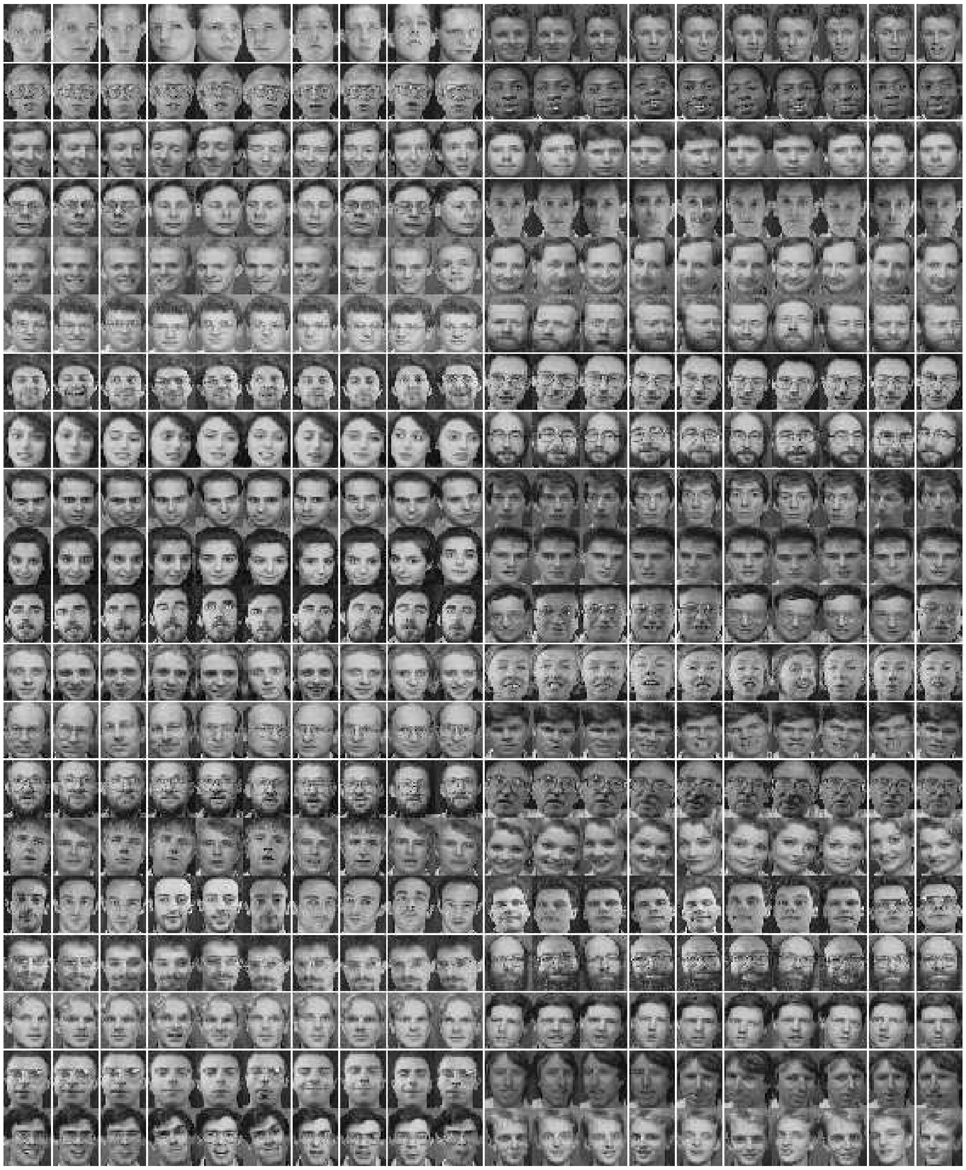

Figure 1: The ORL face database. There are 10 images each of the 40 subjects.

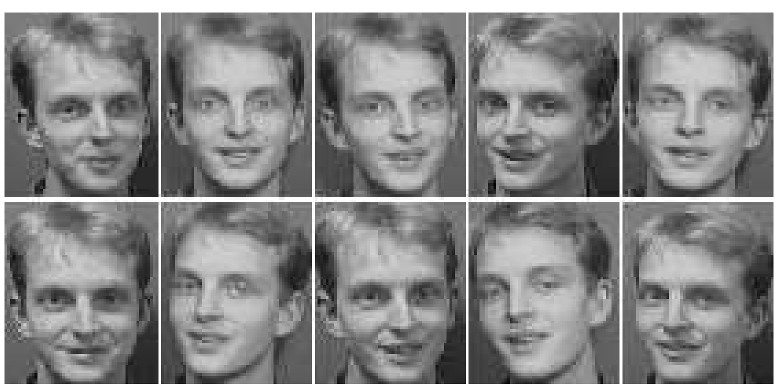

Figure 2: The set of 10 images for one subject. Considerable variation can be seen.

3 Related Work

3.1 Geometrical Features

Many people have explored geometrical feature based methods for face recognition. Kanade [18] presented an automatic feature extraction method based on ratios of distances and reported a recognition rate of between 45-75% with a database of 20 people. Brunelli and Poggio [6] compute a set of geometrical features such as nose width and length, mouth position, and chin shape. They report a 90% recognition rate on a database of 47 people. However, they show that a simple template matching scheme provides 100% recognition for the same database. Cox et al. [9] have recently introduced a *mixture-distance* technique which achieves a recognition rate of 95% using a query database of 95 images from a total of 685 individuals. Each face is represented by 30 *manually* extracted distances.

Systems which employ precisely measured distances between features may be most useful for finding possible matches in a large mugshot database[4]. For other applications, automatic identification of these points would be required, and the resulting system would be dependent on the accuracy of the feature location algorithm. Current algorithms for automatic location of feature points do not consistently provide a high degree of accuracy [41].

3.2 Eigenfaces

High-level recognition tasks are typically modeled with many stages of processing as in the Marr paradigm of progressing from images to surfaces to three-dimensional models to matched models [28]. However, Turk and Pentland [42] argue that it is likely that there is also a recognition process based on low-level, two-dimensional image processing. Their argument is based on the early development and extreme rapidity of face recognition in humans, and on physiological experiments in monkey cortex which claim to have isolated neurons that respond selectively to faces [35]. However, it is not clear that these experiments exclude the sole operation of the Marr paradigm.

Turk and Pentland [42] present a face recognition scheme in which face images are projected onto the principal components of the original set of training images. The resulting *eigenfaces* are classified by comparison with known individuals. The linear principle components technique assumes that the faces lie in a lower dimensional space, and hence the sum or average of two faces should also be a face. Clearly this is not true when principal components is applied to an entire face [17].

Turk and Pentland present results on a database of 16 subjects with various head orientation, scaling, and lighting. Their images appear identical otherwise with little variation in facial expression, facial details, pose, etc. For lighting, orientation, and scale variation their system achieves 96%, 85% and 64% correct classification respectively. Scale is renormalized to the eigenface size based on an estimate of the head size. The middle of the faces is accentuated, reducing any negative affect of changing hairstyle and backgrounds.

In Pentland et al. [34, 33] good results are reported on a large database (95% recognition of 200 people from a database of 3,000). It is difficult to draw broad conclusions as many of the images of the same people look very similar, and the database has accurate registration and alignment [30]. In Moghaddam and Pentland [30], very good results are reported with the FERET database - only one mistake was made in classifying 150 frontal view images. The system used extensive preprocessing for head location, feature detection, and

[4] A mugshot database typically contains side views where the performance of feature point methods is known to improve [8].

normalization for the geometry of the face, translation, lighting, contrast, rotation, and scale.

In summary, it appears that eigenfaces is a fast, simple, and practical algorithm that may be limited due to the requirement that there is a high degree of correlation between the pixel intensities of the training and test images. This limitation has been addressed by using extensive preprocessing to normalize the images.

3.3 Template Matching

Template matching methods such as [6] operate by performing direct correlation of image segments. Template matching is only effective when the query images have the same scale, orientation, and illumination as the training images [9].

3.4 Neural Network Approaches

Much of the present literature on face recognition with neural networks presents results with only a small number of classes (often below 20). For example, in [10] the first 50 principal components of the images are extracted and reduced to 5 dimensions using an autoassociative neural network. The resulting representation is classified using a standard multi-layer perceptron. Good results are reported but the database is quite simple: the pictures are manually aligned and there is no lighting variation, rotation, or tilting. There are 20 people in the database.

3.5 The ORL Database

In [38] a HMM-based approach is used for classification of the ORL database images. The best model resulted in a 13% error rate. Samaria also performed extensive tests using the popular eigenfaces algorithm [42] on the ORL database and reported a best error rate of around 10% when the number of eigenfaces was between 175 and 199. We implemented the eigenfaces algorithm and also observed around 10% error. In [39] Samaria extends the top-down HMM of [38] with pseudo two-dimensional HMMs. The error rate reduces to 5% at the expense of high computational complexity - a single classification takes four minutes on a Sun Sparc II. Samaria notes that although an increased recognition rate was achieved the segmentation obtained with the pseudo two-dimensional HMMs appeared quite erratic. Samaria uses the same training and test set sizes as we do (200 training images and 200 test images with no overlap between the two sets). The 5% error rate is the best error rate previously reported for the ORL database that we are aware of.

4 System Components

4.1 Overview

In the following sections we introduce the techniques which form the components of our system and describe our motivation for using them. Briefly, we explore the use of local image sampling and a technique for partial lighting invariance, a self-organizing map (SOM) for projection of the texture representation into a quantized lower dimensional space, the Karhunen-Loève (KL) transform for comparison with the self-organizing map, a convolutional network (CN) for partial translation and deformation invariance, and a multi-layer perceptron (MLP) for comparison with the convolutional network.

4.2 Local Image Sampling

We have evaluated two different methods of representing local image samples. In each method a window is scanned over the image as shown in figure 3.

1. The first method simply creates a vector from a local window on the image using the intensity values at each point in the window. Let x_{ij} be the intensity at the ith column, and the jth row of the given image. If the local window is a square of sides $2W+1$ long, centered on x_{ij}, then the vector associated with this window is simply $[x_{i-W,j-W}, x_{i-W,j-W+1}, \ldots, x_{ij}, \ldots, x_{i+W,j+W-1}, x_{i+W,j+W}]$.

2. The second method creates a representation of the local sample by forming a vector out of a) the intensity of the center pixel x_{ij}, and b) the difference in intensity between the center pixel and all other pixels within the square window. The vector is given by $[x_{ij} - x_{i-W,j-W}, x_{ij} - x_{i-W,j-W+1}, \ldots, w_{ij}x_{ij}, \ldots, x_{ij} - x_{i+W,j+W-1}, x_{ij} - x_{i+W,j+W}]$. The resulting representation becomes partially invariant to variations in intensity of the complete sample. The degree of invariance can be modified by adjusting the weight w_{ij} connected to the central intensity component.

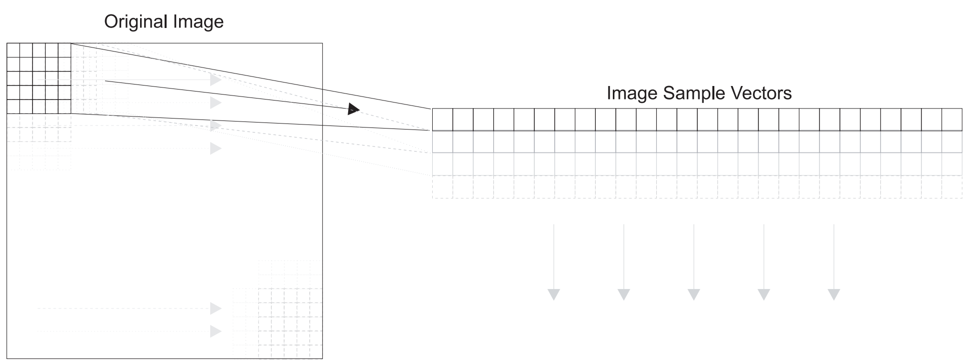

Figure 3: A depiction of the local image sampling process. A window is stepped over the image and a vector is created at each location.

4.3 The Self-Organizing Map

4.3.1 Introduction

Maps are an important part of both natural and artificial neural information processing systems [2]. Examples of maps in the nervous system are retinotopic maps in the visual cortex [32], tonotopic maps in the auditory cortex [19], and maps from the skin onto the somatosensoric cortex [31]. The self-organizing map, or SOM, introduced by Teuvo Kohonen [21, 20] is an unsupervised learning process which learns the distribution of a set of patterns without any class information. A pattern is projected from an input space to a position in the map - information is coded as the location of an activated node. The SOM is unlike most classification or clustering techniques in that it provides a topological ordering of the classes. Similarity in input patterns is preserved in the output of the process. The topological preservation of the SOM process makes it especially useful in the classification of data which includes a large number of classes. In the local image sample classification, for example, there may be a very large number of classes in which the transition from one class to the next is practically continuous (making it difficult to define hard class boundaries).

4.3.2 Algorithm

We give a brief description of the SOM algorithm, for more details see [21]. The SOM defines a mapping from an input space $\mathcal{R}^n$ onto a topologically ordered set of nodes, usually in a lower dimensional space. An example of a two-dimensional SOM is shown in figure 4. A reference vector in the input space, $m_i \equiv [\mu_{i1}, \mu_{i2}, ..., \mu_{in}]^T \in \mathcal{R}^n$, is assigned to each node in the SOM. During training, each input, x, is compared to all of the m_i, obtaining the location of the closest match ($||x - m_c|| = \min_i\{||x - m_i||\}$). The input point is mapped to this location in the SOM. Nodes in the SOM are updated according to:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{1}$$

where t is the time during learning and $h_{ci}(t)$ is the *neighborhood function*, a smoothing kernel which is maximum at m_c. Usually, $h_{ci}(t) = h(||r_c - r_i||, t)$, where r_c and r_i represent the location of the nodes in the SOM output space. r_c is the node with the closest weight vector to the input sample and r_i ranges over all nodes. $h_{ci}(t)$ approaches 0 as $||r_c - r_i||$ increases and also as t approaches ∞. A widely applied neighborhood function is:

$$h_{ci} = \alpha(t) \exp\left(-\frac{||r_c - r_i||^2}{2\sigma^2(t)}\right) \tag{2}$$

where $\alpha(t)$ is a scalar valued learning rate and $\sigma(t)$ defines the width of the kernel. They are generally both monotonically decreasing with time. The use of the neighborhood function means that nodes which are topographically close in the SOM structure activate each other to learn something from the same input x. A relaxation or smoothing effect results which leads to a global ordering of the map. Note that $\sigma(t)$ should not be reduced too far as the map will lose its topographical order if neighboring nodes are not updated along with the closest node. The SOM can be considered a non-linear projection of the probability density, $p(x)$ [21].

4.3.3 Improving the Basic SOM

The original self-organizing map is computationally expensive due to:

1. In the early stages of learning, many nodes are adjusted in a correlated manner. Luttrel [27] proposed a method which we use that starts by learning in a small network, and doubles the size of the network periodically during training. When doubling, new nodes are inserted between the current nodes. The weights of the new nodes are set equal to the average of the weights of the immediately neighboring nodes.

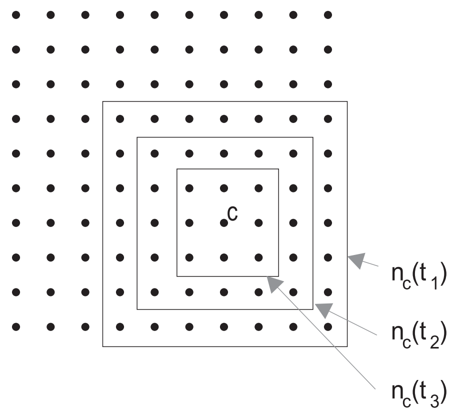

Figure 4: A two-dimensional SOM showing a square neighborhood function which starts as $n_c(t_1)$ and reduces in size to $n_c(t_3)$ over time.

2. Each learning pass requires computation of the distance of the current sample to all nodes in the network, which is $O(N)$. However, this may be reduced to $O(logN)$ using a hierarchy of networks which is created from the above node doubling strategy[5].

4.4 Karhunen-Loève Transform

The optimal linear method[6] for reducing redundancy in a dataset is the Karhunen-Loève (KL) transform or eigenvector expansion via Principle Components Analysis (PCA) [12]. PCA generates a set of orthogonal axes of projections known as the principal components, or the eigenvectors, of the input data distribution in the order of decreasing variance. The KL transform is a well known statistical method for feature extraction and multivariate data projection and has been used widely in pattern recognition, signal processing, image processing, and data analysis. Points in an n-dimensional input space are projected into an m-dimensional space, $m \leq n$. We use the KL transform for comparison with the SOM in the dimensionality reduction of the local image samples. The use of the KL transform here is not the same as in the eigenfaces approach because we operate on small local image samples as opposed to the entire images.

The KL technique is fundamentally different to the SOM method, as it assumes the images are sufficiently described by second order statistics, while the SOM is an attempt to approximate the probability density as shown in Kohonen [21].

4.5 Convolutional Networks

Theoretically, we should be able to train a large enough multi-layer perceptron neural network to perform any required mapping [14], including that required to perfectly distinguish the classes in face recognition. However, in practice, such a system is unable to form the required features in order to generalize to unseen inputs (the class of functions which can perfectly classify the training data is too large and it is not easy to constrain the solution to the subset of this class which exhibits good generalization). In other words, the problem is ill-posed - there is not enough training points in the space created by the input images in order

[5]This assumes that the topological order is optimal prior to each doubling step.
[6]In the least mean squared error sense.

to allow accurate approximation of class probabilities throughout the input space. Additionally, there is no invariance to translation or local deformation of the images [23]. Convolutional networks (CN) incorporate constraints and achieve some degree of shift and deformation invariance using three ideas: local receptive fields, shared weights, and spatial subsampling. The use of shared weights also reduces the number of parameters in the system aiding generalization. Convolutional networks have been successfully applied to character recognition [24, 22, 23, 5, 3].

A typical convolutional network for recognizing characters is shown in figure 5 [24]. The network consists of a set of layers each of which contains one or more planes. Approximately centered and normalized images enter at the input layer. Each unit in a plane receives input from a small neighborhood in the planes of the previous layer. The idea of connecting units to local receptive fields dates back to the 1960s with the perceptron and Hubel and Wiesel's [15] discovery of locally sensitive, orientation-selective neurons in the cat's visual system [23]. The weights forming the receptive field for a plane are forced to be equal at all points in the plane. Each plane can be considered as a feature map which has a fixed feature detector that is convolved with a local window which is scanned over the planes in the previous layer. Multiple planes are usually used in each layer so that multiple features can be detected. These layers are called convolutional layers. Once a feature has been detected, its exact location is less important. Hence, the convolutional layers are typically followed by another layer which does a local averaging and subsampling operation (eg. for a subsampling factor of 2: $y_{ij} = \left(x_{2i,2j} + x_{2i+1,2j} + x_{2i,2j+1} + x_{2i+1,2j+1}\right)/4$ where y_{ij} is the output of a subsampling plane at position i, j and x_{ij} is the output of the same plane in the previous layer). The network is trained with the usual backpropagation gradient-descent procedure [13].

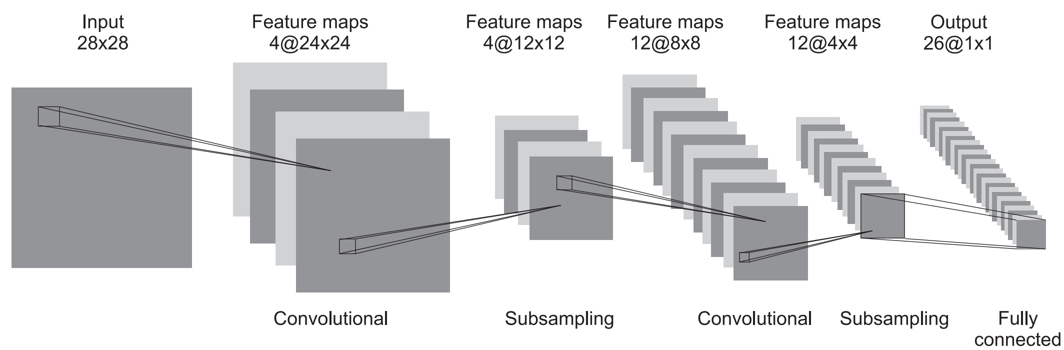

Figure 5: A typical convolutional network for recognizing characters.

5 System Details

The system we have used for face recognition is a combination of the preceding parts - a high-level block diagram is shown in figure 6 and figure 7 shows a breakdown of the various subsystems that we experimented with or discuss.

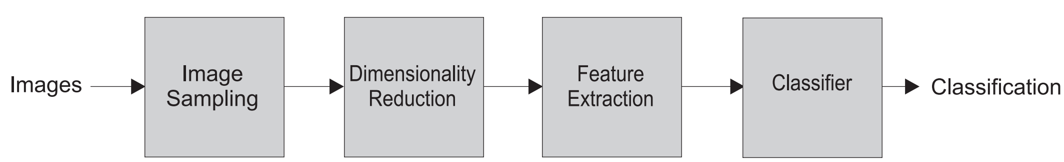

Figure 6: A high-level block diagram of the system we have used for face recognition.

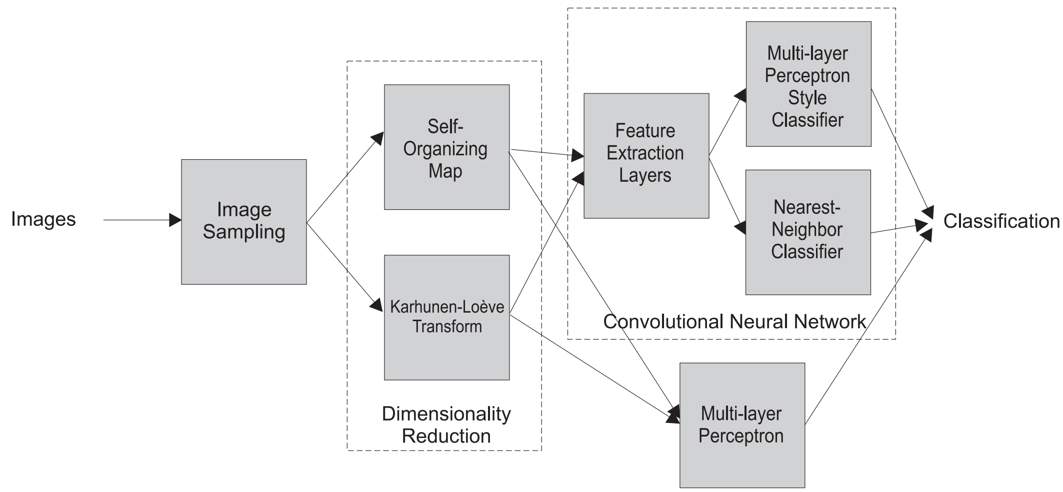

Figure 7: A diagram of the system we have used for face recognition showing alternative methods which we consider in this paper. We present results with either a self-organizing map or the Karhunen-Loève transform used for dimensionality reduction, and either a convolutional neural network or a multi-layer perceptron for classification. We consider the possibility of replacing the final classification stage in the convolutional neural network with a nearest-neighbor or related classifier. A complete recognizer consists of only one path through the diagram.

Our system works as follows (we give complete details of dimensions etc. later):

1. For the images in the training set, a fixed size window (eg. 5x5) is stepped over the entire image as shown in figure 3 and local image samples are extracted at each step. At each step the window is moved by 4 pixels.

2. A self-organizing map (eg. with three dimensions and five nodes per dimension, $5^3 = 125$ total nodes) is trained on the vectors from the previous stage. The SOM quantizes the 25-dimensional input vectors into 125 topologically ordered values. The three dimensions of the SOM can be thought of as three features. We also experimented with replacing the SOM with the Karhunen-Loève transform. In this case, the KL transform projects the vectors in the 25-dimensional space into a 3-dimensional space.

3. The same window as in the first step is stepped over all of the images in the training and test sets. The local image samples are passed through the SOM at each step, thereby creating new training and test sets in the output space created by the self-organizing map. (Each input image is now represented by 3 maps, each of which corresponds to a dimension in the SOM. The size of these maps is equal to the size of the input image (92x112) divided by the step size (for a step size of 4, the maps are 23x28).)

4. A convolutional neural network is trained on the newly created training set. We also experimented with training a standard multi-layer perceptron for comparison.

5.1 Simulation Details

In this section we give the details of one of the best performing systems.

For the SOM, training is split into two phases as recommended by Kohonen [21] - an ordering phase, and a fine-adjustment phase. 100,000 updates are performed in the first phase, and 50,000 in the second. In the first phase, the neighborhood radius starts at two-thirds of the size of the map and reduces linearly to 1. The learning rate during this phase is: $0.7 * (1 - n/N)$ where n is the current update number, and N is the total number of updates. In the second phase, the neighborhood radius starts at 2 and is reduced to 1. The learning rate during this phase is: $0.02 * (1 - n/N)$.

The convolutional network contained five layers excluding the input layer. A confidence measure was calculated for each classification: $y_m(y_m - y_{2m})$ where y_m is the maximum output, and y_{2m} is the second maximum output (for outputs which have been transformed using the *softmax* transformation: $y_i = \frac{\exp(u_i)}{\sum_{j=1}^{k} \exp(u_j)}$ where u_i are the original outputs, y_i are the transformed outputs, and k is the number of outputs). The number of planes in each layer, the dimensions of the planes, and the dimensions of the receptive fields are shown in table 1. The network was trained with backpropagation [13] for a total of 20,000 updates. Weights in the network were updated after each pattern presentation, as opposed to batch update where weights are only updated once per pass through the training set. All inputs were normalized to lie in the range minus one to one. All nodes included a bias input which was part of the optimization process. The best of 10 random weight sets was chosen for the initial parameters of the network by evaluating the performance on the training set. Weights were initialized on a node by node basis as uniformly distributed random numbers in the range $(-2.4/F_i, 2.4/F_i)$ where F_i is the fan-in of neuron i [13]. Target outputs were -0.8 and 0.8 using the $tanh$ output activation function[7]. The quadratic cost function was used. A search then converge learning rate schedule was used[8]: $\eta = \frac{\eta_0}{\frac{n}{N/2} + \frac{c_1}{max\left(1, (c_1 - \frac{max(0, c_1(n - c_2 N))}{(1-c_2)N}\right)}}$ where η = learning rate, η_0 = initial learning rate = 0.1, N = total training epochs, n = current training epoch, $c_1 = 50$, $c_2 = 0.65$. The schedule is shown in figure 8. Total training time was around four hours on an SGI Indy 100Mhz MIPS R4600 system.

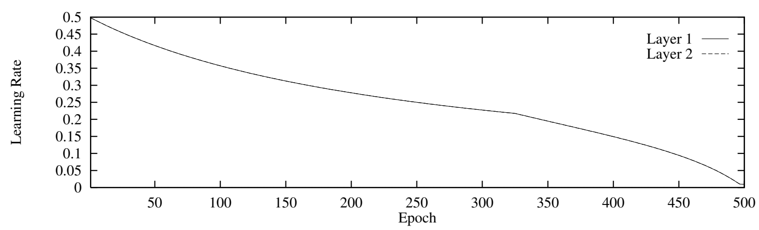

Figure 8: The learning rate as a function of the epoch number.

6 Experimental Results

We performed various experiments and present the results here. Except where noted, all experiments were performed with 5 training images and 5 test images per person for a total of 200 training images and 200 test images. There was no overlap between the training and test sets. We note that a system which guesses the correct answer would be right one out of forty times, giving an error rate of 97.5%. For the following sets

[7] This helps avoid saturating the sigmoid function. If targets were set to the asymptotes of the sigmoid this would tend to: a) drive the weights to infinity, b) cause outlier data to produce very large gradients due to the large weights, and c) produce binary outputs even when incorrect - leading to decreased reliability of the confidence measure.

[8] Relatively high learning rates are typically used in order to help avoid slow convergence and local minima. However, a constant learning rate results in significant parameter and performance fluctuation during the entire training cycle such that the performance of the network can alter significantly from the beginning to the end of the final epoch. Moody and Darkin have proposed "search then converge" learning rate schedules. We have found that these schedules still result in considerable parameter fluctuation and hence we have added another term to further reduce the learning rate over the final epochs. We have found the use of learning rate schedules to improve performance considerably.

Layer	Type	Units	x	y	Receptive field x	Receptive field y	Connection Percentage
1	Convolutional	20	21	26	3	3	100
2	Subsampling	20	9	11	2	2	-
3	Convolutional	25	9	11	3	3	30
4	Subsampling	25	5	6	2	2	-
5	Fully connected	40	1	1	5	6	100

Table 1: Dimensions for the convolutional network. The connection percentage refers to the percentage of nodes in the previous layer which each node in the current layer is connected to - a value less than 100% reduces the total number of weights in the network and may improve generalization. The connection strategy used here is similar to that used by Le Cun et al. for character recognition. As an example of how the precise connections can be determined from the table - the size of the first layer planes (21x26) is equal to the total number of ways of positioning a 3x3 receptive field on the input layer planes (23x28).

of experiments, we vary only one parameter in each case. The error bars shown in the graphs represent plus or minus one standard deviation of the distribution of results from a number of simulations[9]. We note that ideally we would like to have performed more simulations per reported result, however, we were limited in terms of computational capacity available to us. The constants used in each set of experiments were: number of classes: 40, dimensionality reduction method: SOM, dimensions in the SOM: 3, number of nodes per SOM dimension: 5, texture extraction: original intensity values, training images per class: 5. Note that the constants in each set of experiments may not give the best possible performance as the current best performing system was only obtained as a result of these experiments. The experiments are as follows:

1. *Variation of the number of output classes* – table 2 and figure 9 show the error rate of the system as the number of classes is varied from 10 to 20 to 40. We made no attempt to optimize the system for the smaller numbers of classes. As we expect, performance improves with fewer classes to discriminate between (if we continue to add new classes then the chance of a new class being very similar to an existing class increases).

Number of classes	10	20	40
Error rate	1.33%	4.33%	5.75%

Table 2: Error rate of the face recognition system with varying number of classes (subjects). Each result is the average of three simulations.

2. *Variation of the dimensionality of the SOM* – table 3 and figure 10 show the error rate of the system as the dimension of the self-organizing map is varied from 1 to 4. The best performing value is three dimensions.

3. *Variation of the quantization level of the SOM* – table 4 and figure 11 show the error rate of the system as the size of the self-organizing map is varied from 4 to 8 nodes per dimension. The SOM has three dimensions in each case. The best error rate occurs for 8 nodes per dimension. This is also the best error rate of all experiments.

[9] We ran multiple simulations in each experiment where we varied the selection of the training and test images (out of a total of $10!/5! = 30240$ possibilities) and the random seed used to initialize the weights in the convolutional neural network.

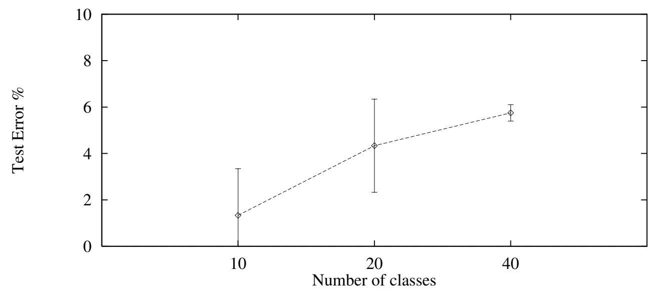

Figure 9: The error rate as a function of the number of classes. We did not modify the network from that used for the 40 class case.

SOM Dimension	1	2	3	4
Error rate	8.25%	6.75%	5.75%	5.83%

Table 3: Error rate of the face recognition system with varying number of dimensions in the self-organizing map. Each result given is the average of three simulations.

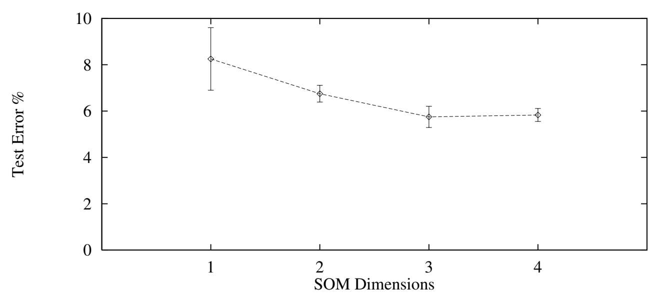

Figure 10: The error rate as a function of the number of dimensions in the SOM.

SOM Size	4	5	6	7	8
Error rate	8.5%	5.75%	6.0%	5.75%	3.83%

Table 4: Error rate of the face recognition system with varying number of nodes per dimension in the self-organizing map. Each result given is the average of three simulations.

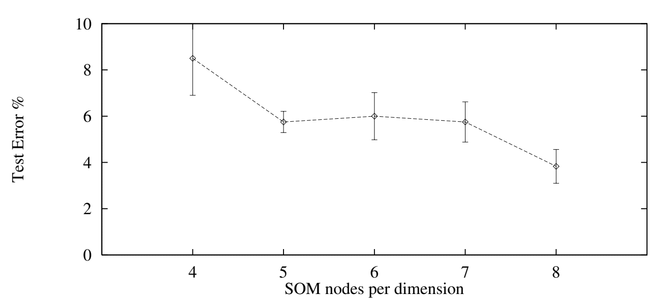

Figure 11: The error rate as a function of the number of nodes per dimension in the SOM.

4. *Variation of the texture extraction algorithm* – table 5 shows the result of using the two local image sample representations described earlier. We found that using the original intensity values gave the best performance. We tried altering the weight assigned to the central intensity value in the alternative representation but were unable to improve the results.

Input type	Pixel intensities	Differences w/base intensity
Error rate	5.75%	7.17%

Table 5: Error rate of the face recognition system with varying image sample representation. Each result is the average of three simulations.

5. *Substituting the SOM with the KL transform* – table 6 shows the results of replacing the self-organizing map with the Karhunen-Loève transform. We tried using the first one, two, or three eigenvectors for projection. Surprisingly, the system performed best with only 1 eigenvector. The best SOM parameters we tried produced slightly better performance. The quantization inherent in the SOM could provide a degree of invariance to minor image sample differences and quantization of the PCA projections may improve performance.

Dimensionality reduction	Linear PCA	SOM
Error rate	5.33%	3.83%

Table 6: Error rate of the face recognition system with linear PCA and SOM feature extraction mechanisms. Each result is the average of three simulations.

6. *Replacing the CN with an MLP* – table 7 shows the results of replacing the convolutional network with a multi-layer perceptron. Performance is very poor, as we expect due to the loss of shift and deformation invariance. We tried a number of different hidden layer sizes for the multi-layer perceptron in the range 20 to 100. Note that the best performing KL parameters were used while the best performing SOM parameters were not.

	Linear PCA	SOM
MLP	41.2%	39.6%
CN	5.33%	3.83%

Table 7: Error rate comparison of the various feature extraction and classification methods. Each result is the average of three simulations.

7. *The tradeoff between rejection threshold and recognition accuracy* – Figure 12 shows a histogram of the recognizer's confidence for the cases when the classifier is correct and when it is wrong for one of the best performing systems. From this graph we expect that classification performance will increase significantly if we reject cases below a certain confidence threshold. Figure 13 shows the system performance as the rejection threshold is increased. We can see that by rejecting examples with low confidence we can significantly increase the classification performance of the system. If we consider a system which used a video camera to take a number of pictures over a short period, we could expect that a high performance would be attainable with an appropriate rejection threshold.

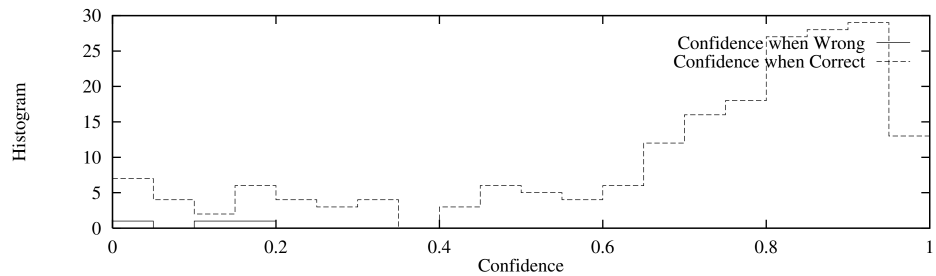

Figure 12: A histogram depicting the confidence of the classifier when it turns out to be correct, and the confidence when it is wrong. The graph suggests that we can improve classification performance considerably by rejecting cases where the classifier has a low confidence.

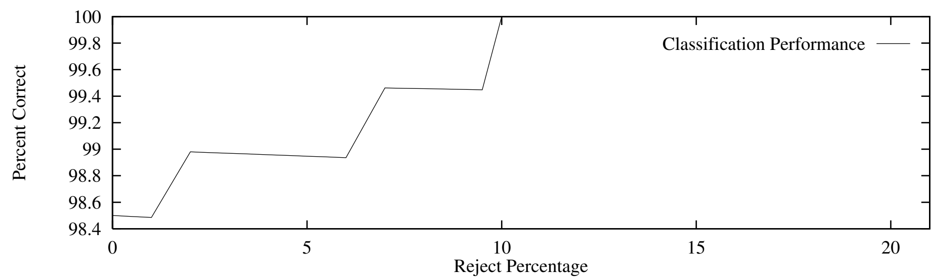

Figure 13: The test set classification performance as a function of the percentage of samples rejected. Classification performance can be improved significantly by rejecting cases with low confidence.

8. *Comparison with other known results on the same database* – Table 8 shows a summary of the performance of the systems for which we have results using the ORL database. In this case, we used a SOM quantization level of 8. Our system is the best performing system[10] and performs recognition roughly 500 times faster than the second best performing system - the pseudo 2D-HMMs of Samaria. Figure 14 shows the images which were incorrectly classified for one of the best performing systems.

System	Error rate	Classification time
Top-down HMM	13%	n/a
Eigenfaces	10.5%	n/a
Pseudo 2D-HMM	5%	240 seconds[1]
SOM+CN	3.8%	< 0.5 seconds[2]

Table 8: Error rate of the various systems. [1] On a Sun Sparc II. [2] On an SGI Indy MIPS R4600 100Mhz system.

9. *Variation of the number of training images per person.* Table 9 shows the results of varying the number of images per class used in the training set from 1 to 5 for PCA+CN, SOM+CN and also for the eigenfaces algorithm. We implemented two versions of the eigenfaces algorithm - the first version creates vectors for each class in the training set by averaging the results of the eigenface representation over all images for the same person. This corresponds to the algorithm as described by Turk and Pentland [42]. However, we found that using separate training vectors for each training image resulted in better performance. We found that using between 40 to 100 eigenfaces resulted in similar performance. We can see that the PCA+CN and SOM+CN methods are both superior to the

[10] The 4% error rate reported is an average of multiple simulations - individual simulations have given error rates as low as 1.5%

Figure 14: Test images. The images with a thick white border were incorrectly classified by one of the best performing systems.

eigenfaces technique even when there is only one training image per person. The SOM+CN method consistently performs better than the PCA+CN method.

Images per person	1	2	3	4	5
Eigenfaces - average per class	38.6	28.8	28.9	27.1	26
Eigenfaces - one per image	38.6	20.9	18.2	15.4	10.5
PCA+CN	34.2	17.2	13.2	12.1	7.5
SOM+CN	30.0	17.0	11.8	7.1	3.5

Table 9: Error rate for the eigenfaces algorithm and the SOM+CN as the size of the training set is varied from 1 to 5 images per person. Averaged over two different selections of the training and test sets.

7 Discussion

Figure 15 shows the randomly chosen initial local image samples corresponding to each node in a two-dimensional SOM, and the final samples which the SOM converges to. Scanning across the rows and columns we can see that the quantized samples represent smoothly changing shading patterns. This is the initial representation from which successively higher level features are extracted using the convolutional network. Figure 16 shows the activation of the nodes in a sample convolutional network for a particular test image.

We use both fixed feature extraction (the representation of local image samples), and a trainable feature extractor (the convolutional network). Can this trainable feature extractor form the optimal set of features? The answer is negative - it is unlikely that the network could extract an optimal set of features for all images. Although the exact process of human face recognition is unknown, there are many features which humans

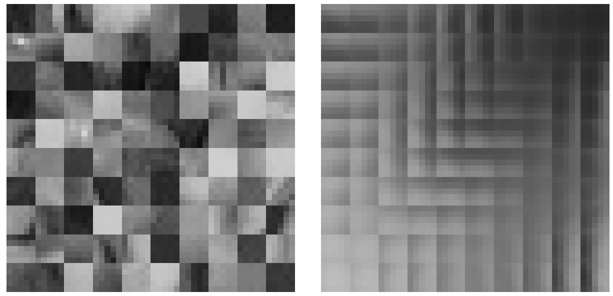

Figure 15: SOM image samples before training (a random set of image samples) and after training.

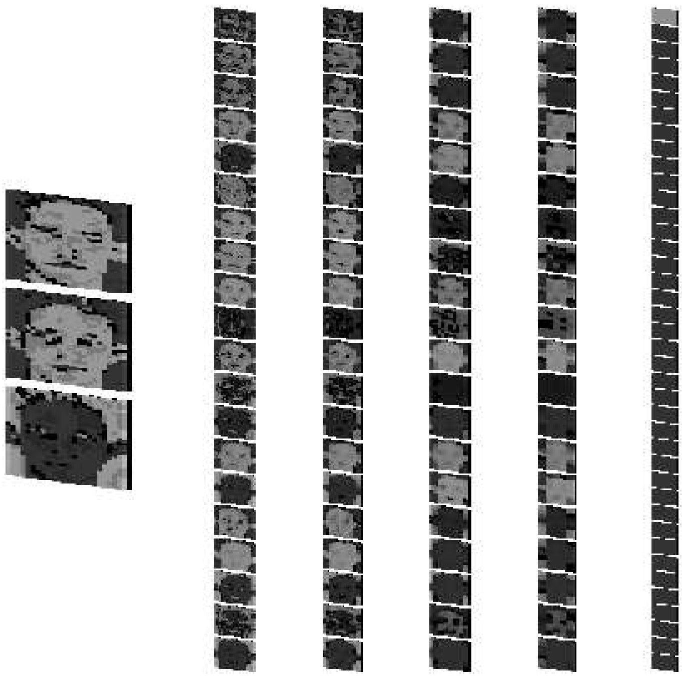

Figure 16: A depiction of the node maps in a sample convolutional network showing the activation values for a particular test image. In this case the the image is correctly classified with only one activated output node (the top node). From left to right, the layers are: the input layer, convolutional layer 1, subsampling layer 1, convolutional layer 2, subsampling layer 2, and the output layer.

may use but our system is unlikely to discover optimally - eg. a) knowledge of the three-dimensional structure of the face, b) knowledge of the nose, eyes, mouth, etc., c) generalization to glasses/no glasses, different hair growth, etc., and d) knowledge of facial expressions.

8 Computational Complexity

The SOM takes considerable time to train. This is not a drawback of the approach however, as the system can be extended to cover new classes without retraining the SOM. All that is required is that the image samples originally used to train the SOM are sufficiently representative of the image samples used in new images. For the experiments we have reported here, the quantized output of the SOM is very similar if we train it with only 20 classes instead of 40. In addition, the Karhunen-Loève transform can be used in place of the SOM with a minimal impact on system performance.

It also takes a considerable amount of time to train a convolutional network, how significant is this? The convolutional network extracts features from the image. Other methods have fixed feature extraction algorithms which are not trained (eg. eigenfaces) - we can do the same here. Consider if we separate the convolutional network into two parts: the initial feature extraction layers and the final feature extraction and classification layers. Given a well chosen sample of the complete distribution of faces which we want to recognize, the features extracted from the first section could be expected to also be useful for the classification of new classes. These features could then be considered fixed features and the first part of the network may not need to be retrained when adding new classes. The point at which the convolutional network is broken into two would depend on how well the features at each stage are useful for the classification of new classes (the larger features in the final layers are less likely to be a good basis for classification of new examples). We note that it may be possible to replace the second part with another type of classifier - eg. a nearest-neighbor classifier. In this case the time required for retraining the system when adding new classes is minimal (the extracted feature vectors are simply stored for the training images).

To give an idea of the computational complexity of each part of the system we define:

N_c	The number of classes
N_s	The number of nodes in the self-organizing map
N_{w1}	The number of weights in the convolutional network
N_{w2}	The number of weights in the classifier
N_{tr}	The number of training examples
N_n	The number of nodes in the neighborhood function
N_{nn1}	The total number of next nodes used to backpropagate the error in the CN
N_{nn2}	The total number of next nodes used to backpropagate the error in the MLP classifier
N_{od}	The output dimension of the KL projection
N_{id}	The input dimension of the KL projection
$N_{samples}$	The number of training samples for the SOM or the KL projection
N_{window}	The number of local image samples per image

Tables 10 and 11 show the approximate complexity of the various parts of the system during training and classification. We show the complexity for both the SOM and KL alternatives for dimensionality reduction and for both the neural network (MLP) and a nearest-neighbor classifier (as the last part of the convolutional network - not as a complete replacement, ie. this is not the same as the earlier multi-layer perceptron experiments). We note that the constant associated with the log factors may increase exponentially in the worst case (cf. neighbor searching in high dimensional spaces [1]).

9 Further Research

We can identify the following avenues for improving performance:

1. More careful selection of the convolutional network architecture, eg. by using the Optimal Brain Damage algorithm [25] as used by Le Cun et al. [24] to improve generalization and speedup handwritten digit recognition.

Section	Training complexity
KL	$O((2+N_{id}^2)N_{samples} + 3N_{od}^3) \approx O(N_{id}^2 + N_{od}^3)$
SOM	$O(k_1 N_{samples} N_n k_2 log N_s) \approx O(N_{samples} N_n log N_s)$ (N_n varies)
CN	$O(k_3 N_{tr}(N_{w1} + N_{nn1})) \approx O(N_{tr} N_{w1})$
MLP Classifier	$O(k_3 N_{tr}(N_{w2} + N_{nn2})) \approx O(N_{tr} N_c)$
NN Classifier	$O(N_{tr})$

Table 10: Training complexity. k_1 and k_3 represent the number of times the training set is presented to the network for the SOM and the CN respectively.

Section	Classification complexity
KL	$O(N_{window} N_{id} N_{od})$
SOM	$O(N_{window} k_1 log N_s) \approx O(N_{window} log N_s)$
CN	$O(k_2 N_{w1}) \approx O(N_{w1})$
MLP Classifier	$O(N_{w2}) \approx O(N_c)$
NN Classifier	$O(k_4 log N_{tr}) \approx O(log N_c)$

Table 11: Classification complexity. k_2 represents the degree of shared weight replication.

2. More precise normalization of the images to account for translation, rotation. Any normalization would be limited by the desired recognition speed.

3. The various facial features could be ranked according to their importance in recognizing faces and separate modules could be introduced for various parts of the face, eg. the eye region, the nose region, and the mouth region (Brunelli and Poggio [6] obtain very good performance using a simple template matching strategy on precisely these regions).

4. An ensemble of recognizers could be used. These could be combined via simple methods such as a linear combination based on the performance of each network, or via a gating network and the Expectation-Maximization algorithm [16, 11]. Examination of the errors made by networks trained with different random seeds and by networks trained with the SOM data versus networks trained with the KL data shows that a combination of networks should improve performance (the set of common errors between the recognizers is often much smaller than the total number of errors).

5. Invariance to a group of desired transformations could be enhanced with the addition of pseudo-data to the training database - ie. the addition of new examples created from the current examples using local deformation, etc. Leen [26] shows that adding pseudo-data can be equivalent to adding a regularizer to the cost function that penalizes changes in the output when the input goes under a transformation for which invariance is desired.

10 Conclusions

We have presented a fast, automatic system for face recognition which is a combination of a local image sample representation, a self-organizing map network, and a convolutional network. The self-organizing map provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, which results in invariance to minor changes in the image samples, and the convolutional neural network provides for partial invariance to translation, rotation, scale, and deformation. Substitution of the Karhunen-Loève transform for the self-organizing map produced similar but slightly worse results. The method is capable of rapid classification, requires only fast, approximate normalization and preprocessing, and consistently exhibits better classification performance than the eigenfaces approach [42] on the database considered as the number of images per person in the training database is varied from 1 to 5. With 5 images per person the proposed method and eigenfaces result in 3.8% and 10.5% error respectively. The recognizer provides a measure of confidence in its output and classification error approaches zero when rejecting as few as 10% of the examples. Training is computationally expensive (around four hours on a MIPS R4600 100Mhz system), however we have shown that retraining of the complete system may not be required in order to add new classes to the recognizer. We have presented avenues for further improvement.

There are no explicit three-dimensional models in our system, however we have found that the quantized local image samples used as input to the convolutional network represent smoothly changing shading patterns. Higher level features are constructed from these building blocks in successive layers of the convolutional network. In comparison with the eigenfaces approach, we believe that the system presented here is able to learn more appropriate features in order to provide improved generalization. The system is partially invariant to changes in the local image samples, scaling, translation and deformation by design.

Acknowledgments

We would like to thank Ingemar Cox for helpful comments and the Olivetti Research Laboratory and Ferdinando Samaria for compiling and maintaining the ORL database.

References

[1] S. Arya and D. M. Mount. Algorithms for fast vector quantization. In J. A. Storer and M. Cohn, editors, *Proceedings of DCC 93: Data Compression Conference*, pages 381–390. IEEE Press, 1993.

[2] Hans-Ulrich Bauer and Klaus R. Pawelzik. Quantifying the neighborhood preservation of Self-Organizing Feature Maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

[3] Y. Bengio, Y. Le Cun, and D. Henderson. Globally trained handwritten word recognizer using spatial representaion, space displacement neural networks and hidden Markov models. In *Advances in Neural Information Processing Systems 6*, San Mateo CA, 1994. Morgan Kaufmann.

[4] J.L. Blue, G.T. Candela, P.J. Grother, R. Chellappa, and C.L. Wilson. Evaluation of pattern classifiers for fingerprint and OCR applications. *Pattern Recognition*, 27(4):485–501, April 1994.

[5] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. Le Cun, U.Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings of the International Conference on Pattern Recognition*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[6] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, October 1993.

[7] D. K. Burton. Text-dependent speaker verification using vector quantization source coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(2):133, 1987.

[8] R. Chellappa, C.L. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740, 1995.

[9] Ingemar J. Cox, Joumana Ghosn, and Peter N. Yianilos. Feature-based face recognition using mixture-distance. Technical report, NEC Research Institute, Princeton, NJ, October 1995.

[10] David DeMers and Garrison Cottrell. Non-linear dimensionality reduction. In S.J. Hanson, J.D. Cowan, and C.L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

[11] Harris Drucker, Corinna Cortes, L. D. Jackel, Y. Le Cun, and Vladimir Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6:1289–1301, 1994.

[12] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition*. Academic Press, Boston, MA, 1990.

[13] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Macmillan, New York, NY, 1994.

[14] K. Hornik, M. Stinchcombe, and H.White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[15] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

[16] R. A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888, 1995.

[17] Xiaoguang Jia and Mark S. Nixon. Extending the feature vector for automatic face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:1167–1176, 1995.

[18] T. Kanade. *Picture Processing by Computer Complex and Recognition of Human Faces*. PhD thesis, Kyoto University, 1973.

[19] Hajime Kita and Yoshikazu Nishikawa. Neural network model of tonotopic map formation based on the temporal theory of auditory sensation. In *Proc. WCNN 93, World Congress on Neural Networks*, volume II, pages 413–418, Hillsdale, NJ, 1993. Lawrence Erlbaum.

[20] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

[21] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, Germany, 1995.

[22] Y. Le Cun. Generalisation and network design strategies. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.

[23] Y. Le Cun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–258. MIT Press, Cambridge, Massachusetts, 1995.

[24] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a backpropagation neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann, San Mateo, CA, 1990.

[25] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, 1990. (Denver 1989), Morgan Kaufmann.

[26] Todd K. Leen. From data distributions to regularization in invariant learning. *Neural Computation*, 3(1):135–143, 1991.

[27] Stephen P. Luttrell. Hierarchical self-organizing networks. In *Proc. 1st IEE Conf. of Artificial Neural Networks*, pages 2–6, London, UK, 1989. British Neural Network Society.

[28] D. Marr. *Vision*. W. H. Freeman, San Francisco, 1982.

[29] B. Miller. Vital signs of identity. *IEEE Spectrum*, pages 22–30, February 1994.

[30] B. Moghaddam and A. Pentland. Face recognition using view-based and modular eigenspaces. In *Automatic Systems for the Identification and Inspection of Humans, SPIE*, volume 2257, 1994.

[31] K. Obermayer, H. Ritter, and K. Schulten. Large-scale simulation of a self-organizing neural network: Formation of a somatotopic map. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers*, pages 71–74, Amsterdam, Netherlands, 1990. North-Holland.

[32] Klaus Obermayer, Gary G. Blasdel, and Klaus Schulten. A neural network model for the formation and for the spatial structure of retinotopic maps, orientation and ocular dominance columns. In Teuvo Kohonen, Kai Mäkisara, Olli Simula, and Jari Kangas, editors, *Artificial Neural Networks*, pages 505–511, Amsterdam, Netherlands, 1991. Elsevier.

[33] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[34] A. Pentland, T. Starner, N. Etcoff, A. Masoiu, O. Oliyide, and M. Turk. Experiments with eigenfaces. In *Looking at People Workshop, International Joint Conference on Artificial Intelligence 1993*, Chamberry, France, 1993.

[35] Perret, Rolls, and Caan. Visual neurones responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, 47:329–342, 1982.

[36] Y.Y. Qi and B.R. Hunt. Signature verification using global and grid features. *PR*, 27(12):1621–1629, December 1994.

[37] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158, School of Computer Science, Cernegie Mellon University, Pittsburgh, PA, July 1995.

[38] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the 2nd IEEE workshop on Applications of Computer Vision*, Sarasota, Florida, 1994.

[39] Ferdinano Silvestro Samaria. *Face Recognition using Hidden Markov Models*. PhD thesis, Trinity College, University of Cambridge, Cambridge, 1994.

[40] Kah-Kay Sung and Tomaso Poggio. Learning human face detection in cluttered scenes. In *Computer Analysis of Images and Patterns*, pages 432–439, 1995.

[41] K. Sutherland, D. Renshaw, and P.B. Denyer. Automatic face recognition. In *First International Conference on Intelligent Systems Engineering*, pages 29–34, Piscataway, NJ, 1992. IEEE Press.

[42] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3:71–86, 1991.