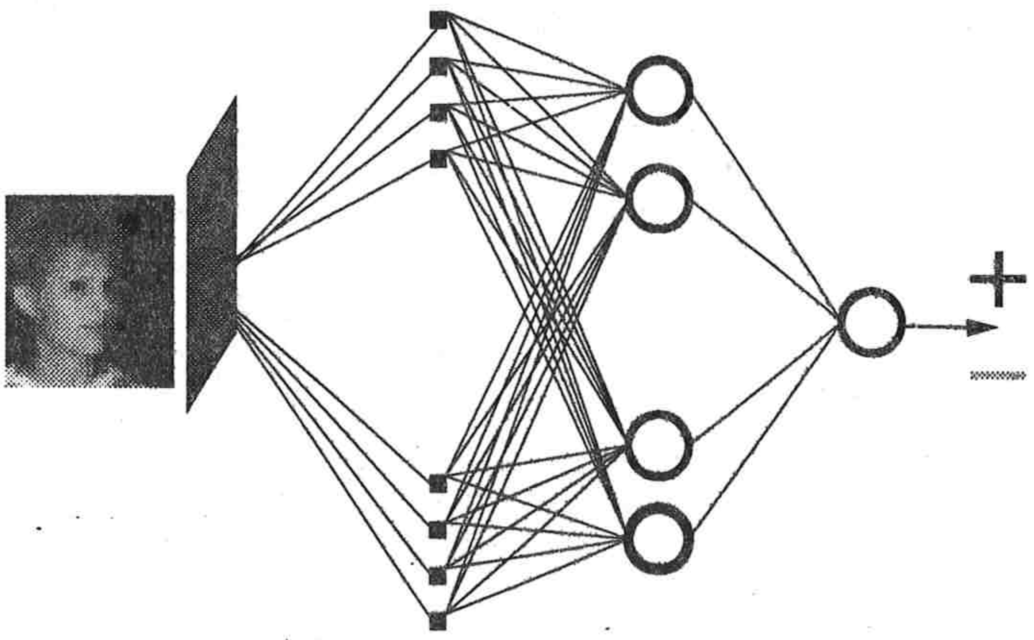
Backpropagation Neural Networks for Facial Recognition

by

Inger Solheim,
University of Tromso, Norway,
and
Los Alamos National Laboratory, Los Alamos, NM, USA



Sivilingeniør Thesis
University of Tromso
December 1991

Abstract

A survey was conducted testing the aptitude of neural networks to recognize human faces. The motivation for this a study is the possibility of designing automatic security systems. Pictures of 511 subjects were collected. The pictures captured both profiles and many natural expressions of the subject. Some of the subjects were wearing glasses, sunglasses or hats in some of the pictures. The images were compressed by a magnitude of 100, and converted into image vectors of 1400 pixels. The image vectors were fed into a back propagation neural network with one hidden layer and one output node. The networks were trained to recognize one target person, and to reject all other persons. Neural networks for 37 target subjects were trained using eight different training sets that consisted of different subsets of the data, and tested on the rest of the data that consisted of 7000 or more unseen pictures. Our results indicate that a false acceptance rate of less than 1% can be obtained, and correct acceptance rate of 98% can be obtained when certain restrictions are followed. A review of earlier work done on face recognition is given, and a brief summary of the theory and methods of neural networks is included.

Foreword

This project was inspired by the need for efficient and accurate personnel verification systems, and by the challenge of using artificial neural networks for solving pattern recognition tasks, with images as patterns.

Several other experiments have been conducted on facial recognition systems. The goal of these systems have been to make systems that yield the identity of a person. The goal of the present project is not to yield the identity of a person, but to verify the identity of a person, based on some a priori knowledge.

The project team consisted of Dr. Ralph Castain, Tanya Payne and myself. Dr. Castain is the formal leader of the project. He supplied the hardware and software for this project. Tanya Panye is a Ph.D. student of cognitive psychology, at the university of New Mexico. Her main interest is in using neural networks as a model of learning processes. She will use this particular project to model how humans recognize faces. My interest and background is in pattern recognition and image processing. Neural networks comprise new and powerful processing techniques for solving tasks in these areas.

The funding of the project started June 1, 1991, and was originally planed to last until the end of December 1991, but because of the positive results of the project, the funding has been extended to last for at least one more year.

Tanya Payne worked two days a week on the project, Ralph Castain provided practical and advisory help whenever necessary. I was the only person who worked full time on the project. The data was collected by Tanya and myself. The first pictures were taken in June and the last pictures were taken in October. The large majority of networks were trained by me, the rest by Ralph Castain or by Tanya Payne. The data analysis in Chapter 8 was carried out by me.

Decisions on training scenarios, and choice of architecture were made through discussions within the project team.

A paper has been submitted, and was recently accepted for the WINN -AIND '92 (Workshop on Neural Networks, Academic/Industrial/NASA/Defence) (*Solheim, Payne and Castain, 1992*), and another paper currently in preparation *(Payne, Solheim, Castain, 1992)*, and will be submitted in Jan 1992 for the IJCNN '92 (International Joint Conference on Neural Networks). The first paper contains a summary of results from four of the training scenarios. The second paper contains a summary of results from eight training scenarios, with an emphasis on the demographic studies we did. A third paper is planned for submittance to Neural Networks. This paper will contain a review of other face recognition projects, together with our results, including a survey of how well the applied methods perform on images of subjects that return over a period of time.

This project is part of an exploding trend of applying neural networks to a variety of different problems. To justify this statement I would like to mention *(Lein, 1991)*, that the estimated budget spent on neural network research in the USA in 1987 was $1 Million, in 1990 this amount passed $150 Millions, and the expected amount spent in 1995 will reach $1.5 Billions. To illustrate the diversity of the field I would like to mention some of the companies that presented papers at the IJCNN conference in Seattle WA, 1991, *(IJCNN, 1991)*: IBM, AT&T Bell Lab., Nippon Telegraph and Telephone, Boeing Commercial Airplane Group, Ford Motor Company, NASA, Mitsubishi, Digital, General Motors, McDonnell Douglas Electronic Systems Co., Toshiba Corp. System and Software Engineering Lab, Institut Francois du Petrole, Siemens AG, Daimler-Benz AG., Olympus Optical Co. Ltd., AEG Comp., Hitachi Ltd., just to mention a few. Of course all the major universities in the USA were represented in addition to universities from France, Britain, Canada, Russia, Japan, China, Taiwan, Denmark, Finland, Germany, Italy, and many other countries. I was curious to find that no one from Norway attended this largest Neural Network Conference. The Swedes were represented by two curious researchers from VOLVO. Hopefully Norwegian industry will pick up on and take advantage of this new and fabulous technology.

Acknowledgments

I would like to thank the following people for their help during this project.

Ralph Castain, for making this project possible and for advising me here in Los Alamos, Torbjorn Eltoft and the University of Tromso, for allowing me to do my diploma work at Los Alamos, and for practical help and advise along the way. Tanya Payne for all the fruitful discussions, helpful comments and good collaboration. Alfred Hanssen and Jan-Erik Solheim for their proofreading and good suggestions. Jonathan Bradley for important discussions on image compression, and for providing the wavelet and cosine compressed images. Bill Saylor for practical help with the computer system, Joe Zowin for patient help with our demographic data base, Grant Nicholas and Vernon Vigil for helping us transport and carry our heavy equipment when we were taking pictures. Last I would like to thank all the 511 enthusiastic people that allowed us to take their pictures, without them there would not have been much of a project!

Contents

1.0 Introduction

There is an increasing need for personnel identity verification devices. The most dominating way of identifying people up to now has been based on some type of possession. This could be an identity card, a key, or a password. These methods have served their purpose, even though they are all fairly easy to fool. False identity cards are flourishing, keys are easy to loose, and many people share their passwords to computer systems or automatic teller machines, with family, friends or colleagues. Many people also get obsessed with finding ways to break such codes. Every day, much damage is being caused by the wrong people having access to computers, buildings or bank accounts.

Because of these and other problems with personnel identification systems, much effort is being done to develope new and more fool proof systems. A class of such systems is based on a set of unique personal biometric features. Some of the ones that have been developed, or are under development are using hand profiles, eye retinal pattern, fingerprints, signature dynamics, finger lengths, and speaker voice characteristics. (Se section 8.5)

Several projects have tried to use the human face as the basis for recognition. This is a natural choice, because the face is one of the most important features used by humans as the basis for identification. The face is however a dynamic, three dimensional structure. It has shown to be very difficult to use traditional pattern recognition techniques to solve this task. (See section 3.5 and 5.1)

With the evolution of artificial neural computing, we have a new method of solving pattern recognition problems. A few studies have been conducted of using neural networks for face recognition. These studies have, to our knowledge, given 100% correct classifications. This is encouraging, but the problem with these studies is that they have been conducted on a very small set of people, far from enough to prove if this is a usable technique over a large population. These surveys have trained the networks to yield the identity of the person facing the camera directly, and not investigated the possibility of recognizing people seen from other angels. Each person in the system is given one class, and when a persons picture is fed into the network, the network will yield a positive output for one of the classes, and zero output for the other classes. This is a nice idea, but impractical, it is also doubtful if it will work for a large population. Each time the recognition device needs to learn a new person, the network needs to relearn all the people in the system. One of the drawbacks with neural networks is that it easily forgets the data it knows, if it is presented with new data, and not the new data together with the old data. Learning is a slow process, and relearning is undesirable. (See chapter 4).

In this project we wanted to investigate the potential for using a neural network in a face recognition system. We wanted to do a large scale experiment which would give a realistic indication of the limitations of the system, how it needs to be designed, and what performance it could yield. To avoid the retraining problem we decided to train many networks that could recognize only one person. The network would learn to recognize one person by being presented with several pictures of that person, and pictures of many other people. The recognition system we imagine, some time in the future, is a system where each person has a set of network weights stored in a magnetic strip on an identity card. The person stands in front of a camera, and the picture of the person is fed into a network with that persons network weights, the person is then recognized or rejected.

For such a system to be a success, it needs to be robust, so that it can recognize people even when daily changes such as glasses or no glasses, shaved or unshaved, and changes in make up and hair style occur. It should also be robust to different positions and angles of the face. One of the goals of this project is to investigate the robustness of the system.

This thesis consists of two parts. The goal of part one is to give the reader an introduction to the theory behind and motivation for choosing, a neural network model to solve the task of recognizing human faces. The first part consist of the following chapters: Chapter 2 that contains a brief review of the history of neural networks. This chapter contains references to some of the most important work done in the field. Chapter 3 contains a short review of the basis for traditional pattern recognition, and how it differs from neural network models. Chapter 4 introduces the theory behind the neural network model we used in the face recognition project. In Chapter 5 a review of papers on face recognition is given. In chapter 6 the data and preprocessing of the data used for the face recognition project is described. Chapter 7 contains a summary of how the network parameters and training scenarios were chosen. Chapter 8 contains a description of the face recognition survey and the results of the survey. In chapter 9 a conclusion and short summary of the project is given. In Appendix A, the program used for pre-processing the image data is included. In Appendix B, The tables of the original result data are included. These form the basis for the analysis done in Chapter 8.

2.0 History of Artificial Neural Networks.

This chapter gives the historic background of neural network technology, and contains references to some of the most important work done in this field.

It is common to take the paper of *Warren McCulloch and Walter Pitts, (1943),* "A logical calculus of the ideas immanent in nervous activity" as the beginning of neuro-computing. This paper showed that simple types of neural networks could in principle compute any arithmetic or logical function. In the 50's Norbert Wiener and John von Neumann, wrote books and papers, *(von Neumann, 1951, 1956)*, suggesting that research into the design of brain-inspired computers might be interesting.

Hebb, (1949), introduced the first learning rule for the neurons in the brain. He made a hypothesis about the way in which synaptic strengths (the communication channels or the links between biological neurons) in the brain change in response to experience. Hebb suggested that the changes were proportional to the correlation between the firing of the pre- and post- synaptic neurons.

The first neuro-computer (Snark) was constructed in 1951 by Marvin Minsky, *(Minsky, 1954)*. Although it didn't work very well it provided design ideas that were used later by other investigators. The perceptron, a parallel distributed processing system made up of a network of multipliers and adding functions (see section 4.2), was invented by Frank

Rosenblatt in 1957, *(Rosenblatt, 1958)*. He built the first successful neuro-computer (the Mark I Perceptron). Rosenblatt studied structures with more layers of units and believed that they could overcome the limitations of the simple perceptron. However, he was not able to find a learning algorithm that could determine the weights necessary to implement any given calculation.

Slightly later Widrow and his graduate students developed a different type of neuro- computer the ADALINE (ADAptive LInear Element), (*Widrow and Hoff, 1960, Widrow, 1962*). Widrow also introduced a powerful learning law that is still widely in use. His learning law attempts to find the best possible set of weight vectors in terms of the least mean square error performance function criterion, *(Widrow and Stearns*, 1985).

Many other scientists had considerable success in developing neuro-computing architectures and implementation concepts. The majority of the research was conducted from a qualitative and experimental point of view rather than from an analytic point of view. The field was suffering of too many dreams and high hopes of constructing artificial brains and truly intelligent machines. By the late sixties people were running out of good ideas, for further progress radically new ideas had to be introduced.

The last episode in neuro-computing's "first era" came with a book by *Minsky and Papert, (1969)*, "Perceptrons". They proved mathematically that the perceptron was unable to perform simple logical operations like the XOR problem. They also doubted that a learning rule could be found to make more than one layer of perceptrons solve this problem. They left the impression that neuro-computing was a dead end, and that other approaches to artificial intelligence would be more profitable. With this conclusion most of the computer science community left the neural network paradigm for almost twenty years.

After this, neuro-computing in the United States suffered a great loss off funding and people, and little explicit neuro-computing research was carried out. However, a great deal of neural network research was conducted in the fields of adaptive signal processing, pattern recognition and biological modeling. In fact much of the work done during these years built the solid platform that the field relies on today. Research in Europe, the Soviet Union and Japan was less affected by the work of Minsky and Papert.

A major theme among those continuing to explore neuro-computing, was associative content - addressable memories. This is a type of network that associates input vectors with similar vectors stored in the network. These were first proposed by *Taylor, (1956)*, and by *Anderson (1968)*. *Kohonen (1974, 1984)*, has been given much credit for advances in this area.

One of the most important innovations in neuro-computing was the backpropagation learning law. This learning law adjusts the weights in a multi layered neural network. The backpropagation neural network is a powerful mapping network which has been successful used in a wide variety of applications. These range from pattern recognition to image compression and predicting outputs of chaotic systems. It is not easy to trace back who originally introduced it. The primary learning law used can be shown to follow from the Robbins/Monroe technique introduced in 1951, *(White, 1989)*. A mathematically similar recursive control algorithm was presented by *Arthur Bryson and Yu-Chi Ho (1975)* in 1969. However Paul Werbos, *(Werbos 1974)*, was given much of the credit for it after he introduced it in 1974. David Rumelhart, Ronald Williams and other members of the PDP group (Parallel Distributed Processing research group, Institute for Cognitive Science, University of California, San Diego) *(Rumelhart et. al, 1986)*, developed the backpropagation into a usable technique, and spread the architecture of it to a large audience.

By the early 1980s many neuro-computing scientists in the United States were again submitting proposals to explore the development of neuro-computing and neural networks applications.

An other important force was the established physicist John Hopfield. He wrote famous papers on neural networks, (i. e. *Hopfield, 1982, 1984*), these papers together with his lectures all around the world recruited hundreds of scientists, mathematicians and technologists to join the emerging field. Hopfield added some helpful insights to the field by introducing energy functions, and by emphasizing the notion of memory as dynamically stable attractors. The learning rules used in the Hopfield model, is the one introduced by *Hebb, (1949).*

In 1986 with the publication of the "PDP" books, "Parallel Distributed Processing", Volume1 and 2, by the PDP group, *(Rumelhart et. al, 1986),* the field exploded. In 1987 the first major open neural networks conference was held after the fields renaissance, the IEEE international conference on neural networks with 1700 participants. In 1988 the INNS (International Neural Network Society) journal Neural Networks was founded. In 1989 the first issue of the journal Neural Computing came out and in 1990 the IEEE Transactions on Neural Networks was first published. Today many universities and institutions have research and educational programs in neuro-computing.

(Much of the content of this section is taken from the books by *Hecht-Nielsen, 1989* and by *Hertz et. al., 1991)*

3.0 Pattern recognition

In this Chapter I will give a short review of some of the traditional pattern recognition techniques and concepts. Section 3.1 contains a brief introduction to pattern recognition, the next section, section 3.2, talks about the challenge in finding and estimating a sufficient set of parameters. Section 3.3 illustrates a training process, and the last section, section 3.4, gives a brief review of the Bayes classifier.

3.1 Traditional Pattern Recognition

The task of pattern recognition is essentially the task of mapping a pattern correctly from a pattern space to a class membership space.

How this process is performed by humans is not fully understood. It is not understood how a person observes an object and then classifies the object, or how a person observes another person, and then recognizes the person as an old friend, or someone who reminds that person of someone he or she hasn't seen for a long time. It's not known what features, or relations between features or something totally different that makes people good pattern recognizer.

The task of traditional computer based pattern recognition is the task of finding such features, relations between features or statistical descriptions of the patterns, that can separate the patterns into classes. Then the task is to quantize them, and describe them to a computer, and in some way or another find the mapping function or algorithm that maps the patterns into the correct class in the class space.

A critical task for traditional computer based pattern recognition is to find the best set of pattern descriptors, and to find the best set of features and relations that yield the most correct mappings in the simplest possible way. Too many features and relations may give very complex decision regions or rules, that may lead to discrimination between patterns that naturally belong together, or may simply give you redundant information. On the other hand, too few features may not discriminate between classes that should be separate. In the case of faces recognition, we want each person to be mapped into its own class. We want to verify that this is Bob, even if he removes his glasses, or shaves his beard off.

The use of adaptive pattern recognition techniques can help selecting the appropriate features needed for the recognition task.

Once the features are selected, there are many ways of designing the pattern classifier. In the following I will give a quick review of the most commonly used techniques.

First there are the parametric models, where one can assume that you know all there is to know about the parameters important to the task. These could be the probability density functions and the probability distributions, and then do the optimal Bayes classification (section 3.4) based on this knowledge. Next there are the non parametric models, i. e. the k-NN (section 3.2), that can be used if there is little or no priori knowledge about the patterns involved in the problem.

3.2 Estimating Parameters

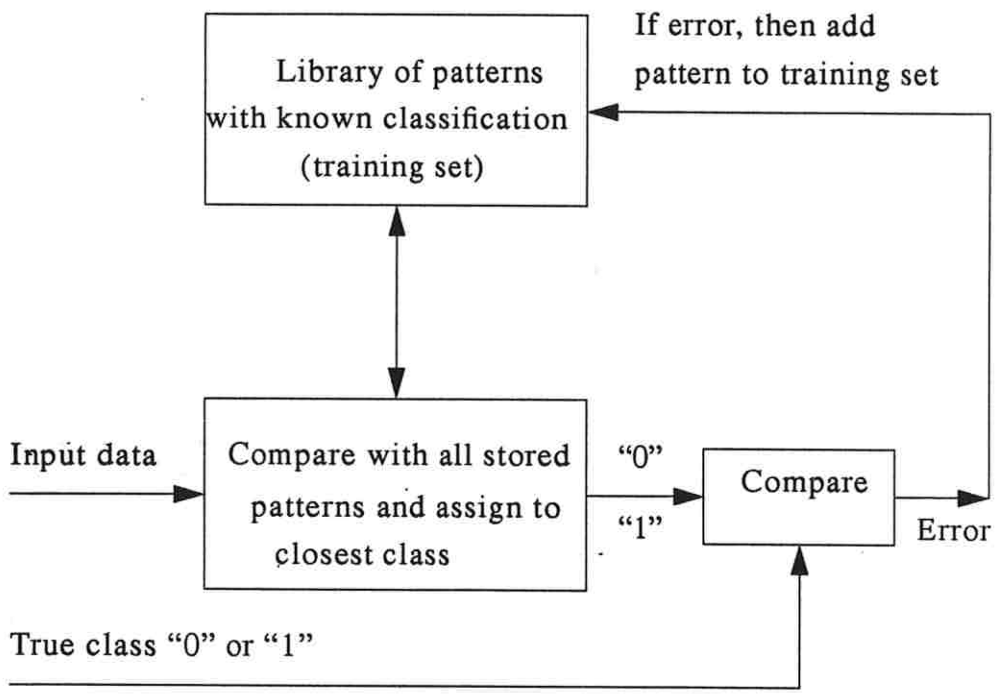
The main challenge in designing a pattern recognition system is to find the necessary and sufficient parameters. The parameters needed to do the classification task are rarely known. Usually one has to use non parametric methods to estimate the probability density functions and distributions. As basis for these one can use Parzen windows, histograms of the data, or the k-NN (k Nearest Neighbors) techniques. *Duba and Hart (1973)* has good descriptions of these methods.

The k-NN techniques are some of the most simple and widely used pattern recognition techniques. This technique assigns the pattern vector $\boldsymbol{x}$ to the same class as, or the same class as the majority of its k nearest neighboring patterns. Nearest or most similar is determined from problem to problem. If the pattern is a vector in space the nearest neighbor can be the pattern in space with the shortest Euclidean distance. If you are working with binary patterns the Hamming distance may be a more natural choice. The Hamming distance between two binary vectors is the number of bits that is different in the two vectors.

3.3 Training the Classifier

When training a classifier you make decision boundaries in the pattern space, or construct a set of decision rules that map the patterns into the right class in the class space. There are two ways of training a pattern classifier. It can be done using supervised or unsupervised learning.

When using supervised learning, the classifier is presented with a set of patterns, and assigned to each pattern is a class membership. Figure 1 is an example of training a k-NN classifier using supervised learning. In this technique one store as many of the training patterns as one needs to classify all the training patterns correctly using the k-NN approach. If the pattern space has a simple structure with well separable classes, one may only have to use the one nearest neighbor, and store only one pattern from each class in the classifier. In general this is not the case, and the number of sample patterns one need to store is usually very large, especially if nothing is known about the probability densities, and the pattern classes are separated in a complex way. Moreover, the demand for a large number of sample patterns grows exponentially with the number of dimensions, which corresponds to the number of features, of the pattern space. This limitation severely restrict the practical application for these nonparametric procedures. More details about this problem can be found in *Duba and Hart, (1973)*.



If error, then add pattern to training set
Library of patterns with known classification (training set)
Input data
Compare with all stored patterns and assign to closest class
"0"
"1"
Compare
Error
True class "0" or "1"

FIGURE 1. Conventional pattern recognition approach that learns by example

When using unsupervised learning, you don't know anything about class membership of the training data you have available. You then use different clustering algorithms, that seek to find clusters or natural groupings in the data. When using these methods you most commonly have to assume that you know certain parameters like the number of clusters you seek to find, the spread or standard deviations in these clusters, and the probability distributions of the clusters. Examples of such methods are the k-mean method, and hierarchical clustering methods. Descriptions of these can be found in *e. g. Duba and Hart, (1973)*, or *Tou and Gonzales, (1974)*.

3.4 The Bayes Classifier

When training of the classifier is carried out, the most desirable way of performing the classification of data with unknown class is to use the principles of the Bayes classifier.

The Bayes classifier is optimal in the sense that it is a minimum risk classifier, *(Tou and Gonzales, 1974)*. The Bayes classifier works in the following manner: Assume that $p(\boldsymbol{x}|\omega_i)$ the conditional probability density function for $\boldsymbol{x}$, the probability density function given that the class is ω_i, is known, and that $P(\omega_i)$ the a priori probability of ω_i occurring is known. Then by Bayes rule:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})} \tag{1}$$

where

$$p(\boldsymbol{x}) = \sum_{j=1}^{M} p(\boldsymbol{x}|\omega_j)P(\omega_j) \tag{2}$$

Bayes rule shows how observing the value of $\boldsymbol{x}$ changes the a priori probability $P(\omega_i)$ to the a posteriori probability $P(\omega_i|\boldsymbol{x})$.

Let L_{ij} be the loss or cost if the classifier assigns $\boldsymbol{x}$ to class ω_j when it actually belongs to class ω_i. Since $\boldsymbol{x}$ may belong to M different classes, the expected loss due to assigning pattern $\boldsymbol{x}$ to the wrong class ω_j is given by

$$r_j(\boldsymbol{x}) = \sum_{i=1}^{M} L_{ij}P(\omega_i|\boldsymbol{x}) = \frac{1}{p(\boldsymbol{x})}\sum_{i}^{M} L_{ij}p(\boldsymbol{x}|\omega_i)p(\omega_i) \tag{3}$$

Since $p(\boldsymbol{x})$ is a common factor we let

$$r_j(\boldsymbol{x}) = \sum_{i}^{M} L_{ij}p(\boldsymbol{x}|\omega_i)p(\omega_i) \tag{4}$$

The minimum risk decision rule for the Bayes classifier then becomes: Assign pattern $\boldsymbol{x}$ to class ω_i if $r_i(\boldsymbol{x}) < r_j(\boldsymbol{x})$ for all i, j; $i \neq j$

If a face recognition system was to be used as a security system for letting people into a secured area, the risk assigned to letting the wrong person into the area would be much grater than the risk of denying the right person entrance. In such a system one would prefer making further investigations of a persons identity, than to risk letting the wrong person in. If however a face recognition system was to be used for finding missing children, the cost of rejecting the right child would be much higher than investigating every one that has the slightest probability of being the right one.

3.5 Face Recognition Using Conventional Techniques

A review of some of the work that has been done on face recognition using conventional techniques is given in section 5.1.

The major problems researchers has met doing face recognition using conventional techniques are the following:

- Finding the best set of features
- Locating and extracting features
- Robustness towards things like glasses, hats, and facial hair
- Dimensionality problems
- Getting enough samples, storing and working with them
- Robustness towards viewing a face from a different angle
- Robustness towards location of the subject in the picture.

To solve the first problem, the solution has been, like in many other pattern recognition problems, that you test out a variety of different features, and the ones you end up with are tailored to give good results on the exact limited population you are working with. The features of the face are three-dimensional, dynamic features. Some of the features in the face, like the mouth and eyes, constantly change with changing facial expressions. Other features that are more stable like the nose, and the distance between the corner of the eye and the ear lobe, are hard to measure exact in a two dimensional picture. The relation between this distance and other static measurements depend very much upon the angle of the face in the image. The last problem with finding and extracting features is the robustness towards natural day to day changes like wearing glasses, or change in hairstyle and make up. To cope with these changes the complexity of the problem becomes even more undesirable.

One could then forget about the extraction of certain features and use the whole image, in color or grey scale, and do a form of template matching *(Jain, 1989, chapter 9)*. Then you immediately meet the dimensionality problem, since a digitized video image is typically of the size 512 X 512 pixels or units. This makes an image vector of 246 144 dimensions. Of course there are many excellent image compression techniques, so you can reduce the dimensionality dramatically, but reducing it to a tractable size, lets say about 100 - 300 dimensions, without losing most of the classification information, is still not a trivial task.

Retrieving enough data samples, i. e. pictures of people‘s faces, should not be to difficult using video cameras that can take “continuous” pictures of a persons face, from all the possible angles. But still the digitized images must be stored before they can be used for computations, and digitized images take a lot of storage space. Which means that using non parametric classification methods where you must store a large amount of digitized images, retrieve them fast, and do calculations and comparisons with them, still will take too much time, and the cost of the storage space and computer power will be unfeasible.

4.0 Neural Networks

This chapter gives a short summary of the theory behind neural networks. First the difference between computations done by neural networks and programmed computers is noted. Then in section 4.1 a definition of neural network models is provided. In section 4.2 one of the earliest building blocks of neural networks, the perceptrons are described. In section 4.3 it is described what can be achieved by combining perceptrons in layers, that is to let the output of a perceptron serve as the input to other perceptrons. Section 4.4 looks at a general neural network, and describes the different types of transfer functions used in the processing elements of the neural networks, and the three most commonly used network types. Then in section 4.5 the notation and derivation of the backpropagation learning law (27) is given. This chapter concludes with demonstrating that the output of the network is in fact an estimate of the a posteriori probability $P(\omega_i | x)$, the a posteriori probability of class ω_i, given that the pattern is **x**.

Research in neuro computing is largely motivated by the possibility of making computing artificial neural networks. Neural networks are well suited to solve tasks like pattern recognition, associative memory and approximation of functions. The term neural networks, however, implies that neuro-computing was originally, and is still for many a way of modeling networks of biological neurons in the brain. The models are extremely simplified when seen from a neuro-psychological point of view, though they are still valuable for gaining insight into the principles of biological computation.

Neuro computing is a fundamentally new and different approach to information processing. It is the first alternative to programmed computing, which has dominated information processing for the last 45 years.

Neuro computing is based on transformations, whereas programmed computing is based on algorithms and rules. Neuro computing is a parallel process, whereas programmed computing is typically a sequential process. The two ways of information handling is computational compatible, but conceptually incompatible. Even if one can to some degree analyze neural networks, find error bounds, robustness, the ability to generalize and to implement certain types of information, these analyzes only assure that neural networks can perform a job, not conceptually how it does it. The simple neural networks models can be analyzed and understood using mathematical methods, but most of the networks are highly nonlinear, and therefore retain a degree of unsolvability or even unpredictably. This implies that unless every possible input is tried in a specific network there is no way to be certain of the exact output of the network. This indicates that a neural network will sometimes make errors, even when they are functioning correctly. This is not the case in programmed computing.

Artificial neural networks are a parallel distributed form of processing and when used for pattern recognition it is an adaptive form of pattern recognition.

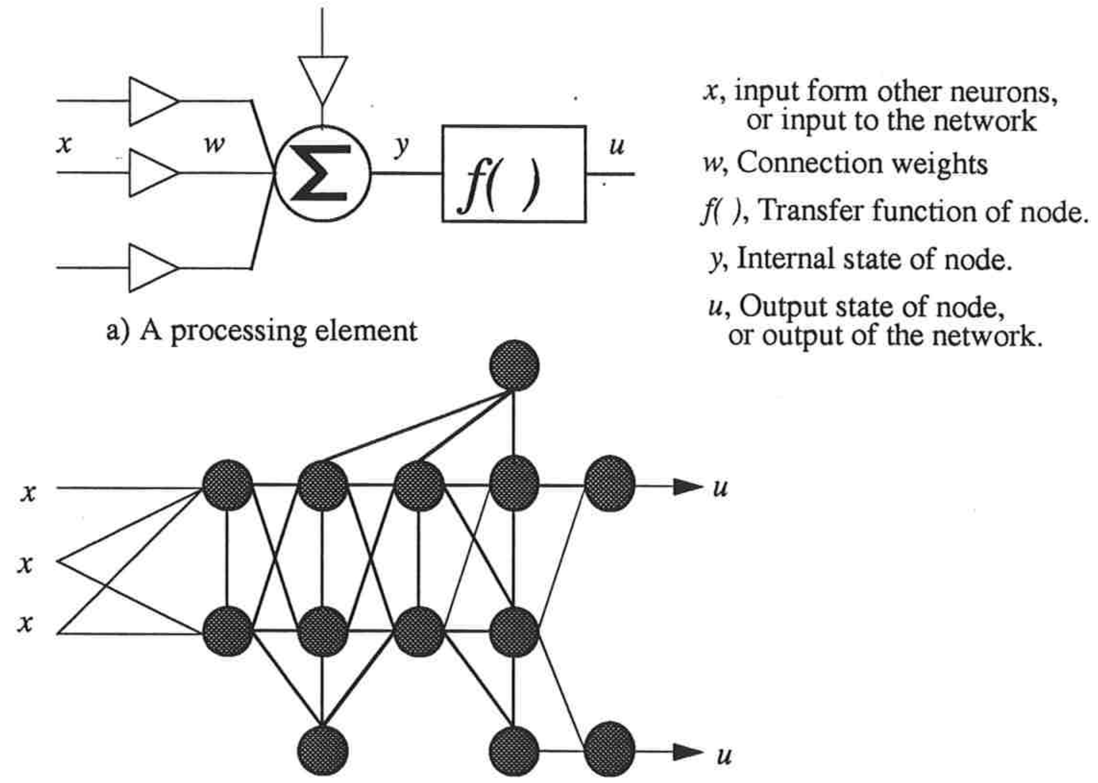
4.1 Neural Networks Models

In the book by *Hecht-Nielsen, (1989)*, Chapter 2 we find a concise definition and description of a neural network. Some parts of the definition are included below.

A neural network is a parallel distributed information processing structure in the form of a directed graph. (A directed graph is a set of points called nodes, along with a set of line segments called links, connecting the nodes). Further definitions concerning neural networks are:

- The nodes in the graph are processing elements
- The links in the graph are called connections. Each connection is an instantaneous unidirectional signal conductions path. Each connection can store a weight, with which the signal transmitted through that path can be multiplied. Each processing unit can receive any number of incoming connections.
- Each processing element can only have one outgoing connection, that can branch out into multiple copies of the outgoing signal. (This is called "fan-out").
- Processing elements can have local memory.
- Each processing element possesses a transfer function, which can use and alter the local memory, use the incoming signals, and which produces the outgoing signal.
- The inputs to a network arrives via connections to the outside world, and the output of a network leave the network through connections out of the network.

Figure 2 a), shows a processing element and Figure 2 b,) shows a general neural network architecture with five layers. There are two conventions in use for counting the number of layers in the networks; some authors count the input terminals as a layer, some do not. We have choose not to count the input terminals as a layer, and only count the layers of processing units as layers. Most network topologies consist of one or more layer of neurons. All networks have connections between layers, but not necessarily fully connected. (A fully connected network is a network where all the nodes in one layer connected to all the nodes in the next layer). Some nets also have connections within a layer.


x
w
y
f( )
u
x, input form other neurons, or input to the network
w, Connection weights
f( ), Transfer function of node.
y, Internal state of node.
u, Output state of node, or output of the network.
x
x
x
u
u

a) A processing element

b) A partially connected network architecture, with five layers.

FIGURE 2. A neural network and its processing element

4.2 The Perceptron

One of the first neural network models was the perceptron invented by Rosenblatt in 1957, *(Pao, 1989 Chapter 5)*. The perceptron is a single processing unit, consisting of a summation operation and a hard limiter. Figure 3 shows a figure of the perceptron. The perceptron has an input consisting of an n+1 dimensional vector, $\boldsymbol{x} = [x_0, x_1, \ldots, x_n]$, where x_0 is permanently set to 1, and is called a bias input. The output of the perceptron is given by

$$y' = \begin{cases} 1, \text{ when } \sum_{i=1}^{N} x_i w_i > T \\ 0, \text{ when } \sum_{i=1}^{N} x_i w_i \leq T \end{cases}, T \text{ is a threshold} \quad (5)$$

The goal of a perceptron is to separate two classes of n-dimensional patterns by an n-1 dimensional hyperplane. This is done by finding a set of weights or adaptive coefficients $(w_0, w_1, \ldots, w_n)$, which determines a hyperplane such that the output of the perceptron is 1 if the input pattern vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)$ belongs to class C_1 and 0 if $\boldsymbol{x}$ belongs to class C_0. This is of course only possible if the classes are linearly separable.

The weights are stored within the processing element and are automatically modified by the processing element itself, according to the perceptron learning law given by

$$w(k+1) = w(k) + (y - y')\,\boldsymbol{x} \qquad (6)$$

where k is the iteration number, y is the desired output of the perceptron, and y' is the actual output of the perceptron given by (5).

During training the perceptron is presented with randomly selected patterns $\boldsymbol{x}$, one at the time, together with the class that $\boldsymbol{x}$ belongs to. On each trial the weights in the perceptron are modified, according to the perceptron learning law given by (6).

The learning process that takes place is such that if the output of the perceptron is wrong when presented with $\boldsymbol{x}$, the weights will be adjusted so that the hyperplane is reoriented, and the output of the perceptron will be correct the next time $\boldsymbol{x}$ or a pattern close to $\boldsymbol{x}$ is presented.

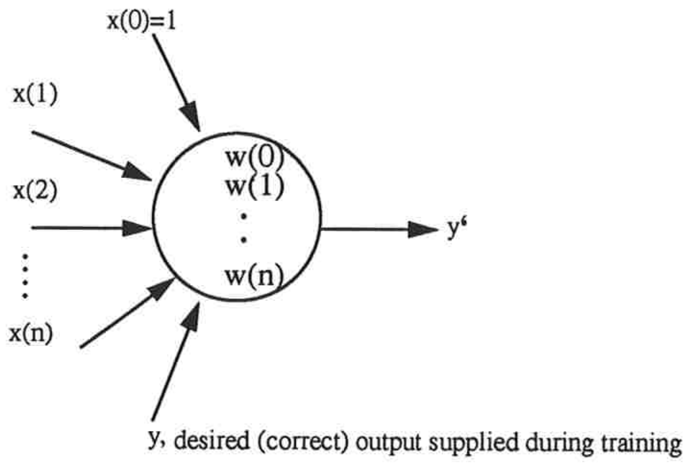


FIGURE 3. The perceptron

4.3 Multi Layered Perceptrons

The single perceptron is a simple processing unit that can only separate linearly separable classes. In real world problems this is most often not the case. By using more than one perceptron, and combining them in layers, that is, letting the output of the perceptron serve as inputs to other perceptrons, it can be shown that any shape or function can be approximated. This is illustrated in figure 4 by *Lippmann, (1987)*. The figure illustrates what kind of decision regions can be made using hard limiting processing units given by (13) for a two class problem in two dimensions.

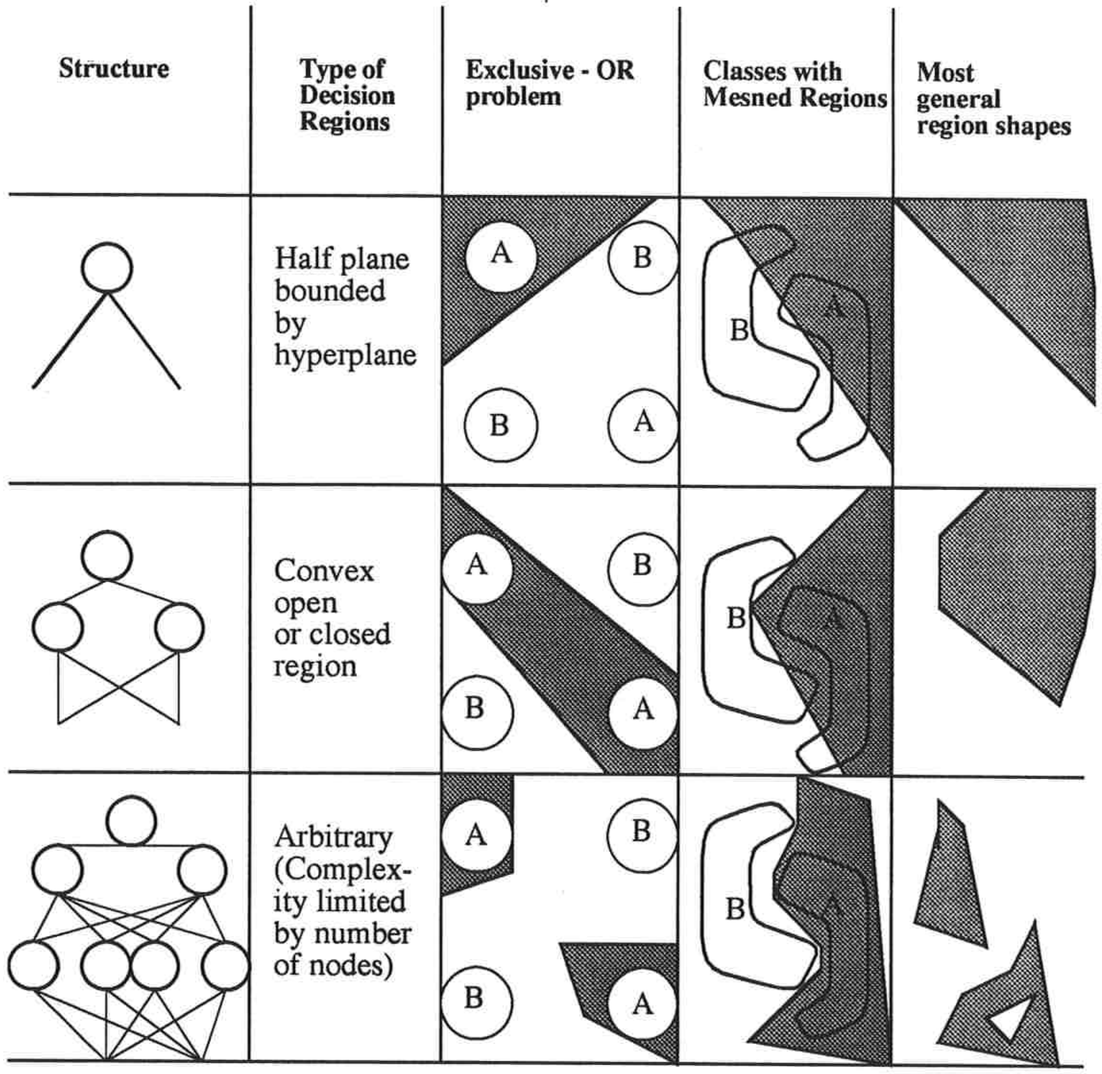

Structure
Type of Decision Regions
Exclusive - OR problem
Classes with Mesned Regions
Most general region shapes
Half plane bounded by hyperplane
Convex open or closed region
Arbitrary (Complexity limited by number of nodes)
A
B

FIGURE 4. **Types of decision regions made by the perceptron (** ***Lippermann 1987*****), In this figure the layers are vertical, instead of horizontal as in the figures in this thesis.**

From figure 4 we see that to map an arbitrary decision boundary we need a three layer network, if we use hard limiters, given by (13), as transfer functions. However it has been shown by *Funahashi (1989),* that if you use a two layer network with a sigmoid activation function given by (14), in the hidden layer, and linear activation function given by (15) in the output layer, then you can form an arbitrary close approximation to any continuous mapping or decision boundary. To approximate any arbitrary, discontinuous decision boundaries, a network with two hidden layers is required.

4.4 A General Neural Network

Let's look at a general neural network, as the one in Figure 2. We can describe the behavior of each neuron, and each layer in a network with a few simple equations. The output of a node *j* is given by y_j:

$$y_j = \sum_{i=1}^{N} w_{ji}x_i + w_{j0} \tag{7}$$

$$u_j = f(y_i) \tag{8}$$

where w_{ji} is the strength of the connection between node *j* and node *i*, and u_j is the output of the node, given by any transfer function $f(y)$. Any layer can be described by:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \tag{9}$$

$$\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{y}) \tag{10}$$

where

$$\boldsymbol{y}^T = [y_1, y_2, \ldots y_N] \tag{11}$$

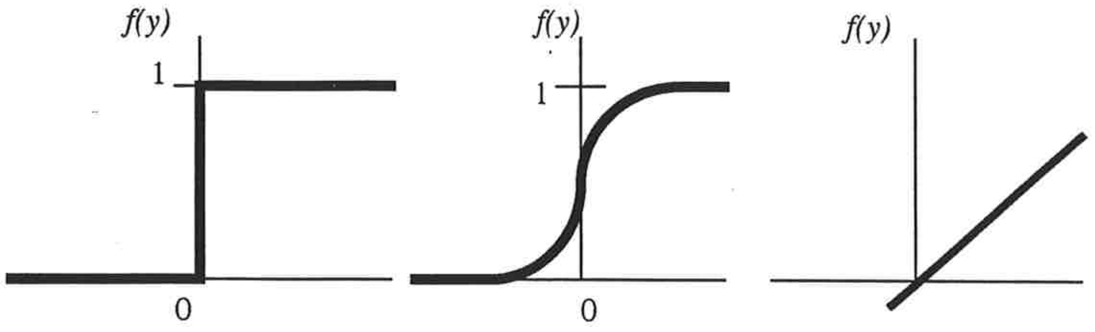
$$\boldsymbol{x}^T = [x_1, x_2 \ldots x_N] \tag{12}$$

The transfer function of the nodes can be arbitrary, but in practice there are three functions that dominate, the hard limiter given by (13) and the sigmoid non linearity given by (14) and the linear function given by (15). A sketch of these functions are given in figure 5. The sigmoid is used for most practical purposes, but because of its shape, it is hard to analyze exactly what is going on in a network using this activation function. For analyzing the internal state of the networks the hard limiter or the linear function are mostly used. The hard limiter can not be used in the learning algorithms for networks with more than one layer, since it has a discontinuous first derivative. If the linear function is used for learning purposes it leads to domination by a single node. Some networks use different transfer functions in different nodes or layers of the network.

$$f_{HL}(y) = \left(\begin{matrix} 1, y > T \\ 0, y \le T \end{matrix}\right) \text{, } T \text{ is a threshold} \tag{13}$$

$$f_s(y) = (1 + e^{-gy})^{-1} \tag{14}$$

$$f_L(y) = y \tag{15}$$



a) Hard limiting non linearity, T=0 b) Sigmoid non linearity c) Linear.

FIGURE 5. Activating functions

The sigmoid transfer function has several desirable properties. It is continuous and varies monotonically from 0 to 1, as y varies from $-\infty$ to ∞. The gain, g determines the steepness of the transition region; if g approaches ∞, the sigmoid approaches the hard limiting function.

We have now looked at the general components of a neural network. By organizing the nodes and connections, and adjusting the weights in various manners, we get networks that can perform different tasks. There exists hundred of different networks architectures, designed by different people, and we can separate the networks roughly into three main categories.

- **Associative memory**. A network whose output is a clean version of a distorted or noisy input pattern. The network "stores" a set of known patterns in its weights, that is the weights model a surface, with attractors. Thus when a pattern in pattern space is fed into the network, the solution, or the output of the network will converge to a specific point or attractor in the pattern space. Each pattern stored in the network, is represented by an attractor. When a new pattern is presented to the network, the network stabilizes in one of its attractors, and we say that the network has associated the input pattern with one of the stored patterns, (*Hertz et al. 1991,* Chapter 2 and 3.)
- **Self organizing feature maps**. These are networks that looks for similarities in the patterns, and organizes them like an unsupervised pattern classifier, *(Hecht-Nielsen, 1989), Chapter 5.4*. The other use of this type of network are networks that search for optimal or near optimal solutions for optimization problems. An example of such a network can be found in *Angeniol et. al., (1988)*.
- **Classifier networks**. Networks that learns by example, *(Rumelhart et. al, 1986, Chapter 8.)*

Since it is the classifier network we used for this project, I will give details only about this type of network. Figure 6 illustrates the learning process of such a network.

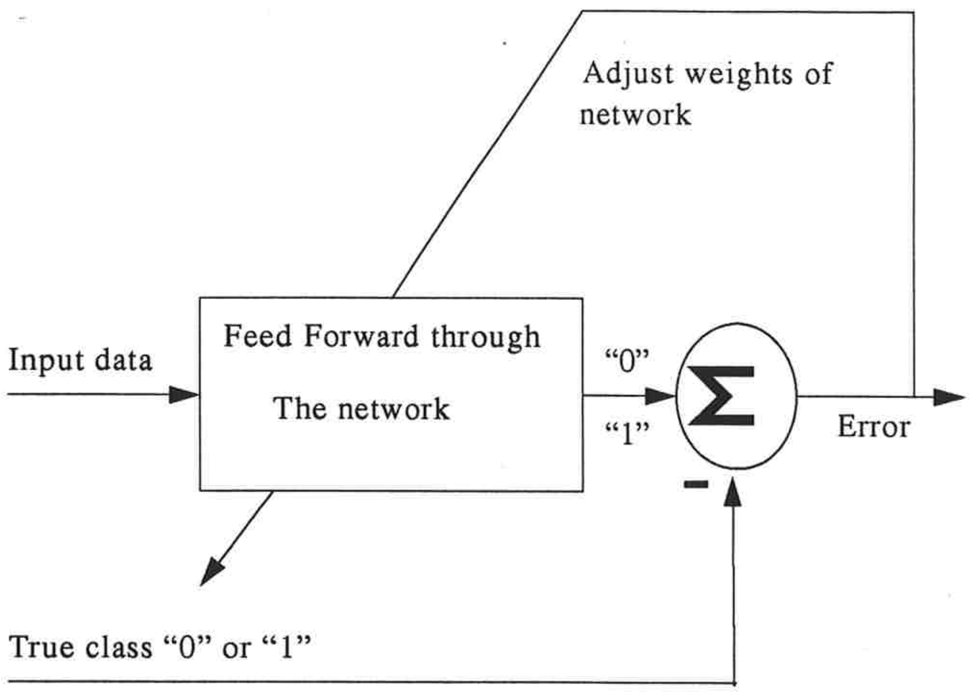


FIGURE 6. Neural network that learns by experience

The characteristic of the network learning process can be summarized in the following way.

- The same data must be shown to network repeatedly during training. This can be a slow process.
- After training, the classification information is stored in the weights of the network. This means that we don't have to store and retrieve the data.
- The computational complexity is constant, and not a function of the number of data samples.
- When learning new data, we must show the network both the new and the old data, or else the network may "forget" the old data.

4.5 The Back Propagation Neural Network

In the following section I will derive the back propagation learning algorithm for the general case with an arbitrary number of hidden layers.

The learning procedure involves presenting the network with a set of pairs of input and output patterns. The system first uses the input pattern to produce an output pattern. Then it compares the output pattern with the desired output. If there is no difference, no learning takes place. If there is a difference, the weights in the network are changed to reduce the difference.

The following notation, taken from *Hush and Horne, (1991)*, will be used.

TABLE 1. Notation

Notation	Meaning
$u_{l,j}$	Output of the j'th node in layer l
$w_{l,j,i}$	the weight which connects the i'th node in layer $l-1$ to the j'th node in layer l
W	the weight matrix, with elements $w_{l,j,i}$
$\boldsymbol{x}_p$	the p'th training sample
$u_{0,i}$	the i'th component of the input vector
$d_j(\boldsymbol{x}_p)$	the desired response of the j'th output layer node for the p'th training sample
N_l	the number of nodes in layer l
L	the number of layers
P	the number of training patterns
ω_i	class i
μ	learning rate

For simplicity let the 0'th layer of the network hold the input vector components, that is $u_{0,j} = x_j$, where x_j is the j'th component of the input vector. The output of a node in layer l is then given by:

$$u_{l,j} = f\left(\sum_{i=1}^{N_{l-1}} w_{l,j,i} u_{l-1,i}\right) \tag{16}$$

where $f(\ldots)$ is the sigmoid non linearity given by (14). This function has a simple derivative

$$f'(y) = \frac{d}{dy}f(y) = gf(y)(1-f(y)) \tag{17}$$

The learning algorithm for the multilayer perceptron uses a gradient search technique, *(Widrow and Strearns, 1985)*. The criterion function to be minimized by the algorithm, is the sum of square error criterion function defined by

$$J(w) = \sum_{p=1}^{P} J_p(w) \tag{18}$$

where $J_p(w)$ is the total square error for the p'th pattern.

$$J_p(w) = \frac{1}{2}\sum_{q=1}^{N_l} (u_{L,q}(x_p) - d_q(x_p))^2 \tag{19}$$

The weights in the network are updated iteratively according to

$$w_{l,j,i}(k+1) = w_{l,j,i}(k) - \mu \left.\frac{\partial J(W)}{\partial w_{l,j,i}}\right|_{W(k)} \tag{20}$$

$$= w_{l,j,i}(k) - \mu \sum_{p=1}^{P} \left.\frac{\partial J_p(W)}{\partial w_{l,j,i}}\right|_{W(k)}$$

Where μ is the learning rate, a small positive number which determines how big steps the gradient decent method will take. To implement the algorithm we must develope an expression for the partial derivative of J_p with respect to each weight in the network. For an arbitrary weight in layer l, this can be written as:

$$\frac{\partial J_p(w)}{\partial w_{l,j,i}} = \frac{\partial J_p(w)}{\partial u_{l,j}} \frac{\partial u_{l,j}}{\partial w_{l,j,i}} \tag{21}$$

where

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i}} = \frac{\partial}{\partial w_{l,j,i}}\left[f\left(\sum_{m=1}^{N_{ll-1}} w_{l,j,m}u_{l-1,m}\right)\right] \tag{22}$$

$$= f'\left(\sum_{m=1}^{N_{l-1}} w_{l,j,i}u_{l-1,m}\right)\frac{d}{dw_{l,j,i}}\left[\sum_{m=1}^{N_{l-1}} w_{l,j,m}u_{l-1,m}\right]$$

using the derivative form (17)

$$\frac{\partial u_{l,j}}{\partial w_{l,j,i}} = u_{l,j}(1-u_{l,j})\frac{\partial}{\partial w_{l,j,i}}\left[\sum_{m=1}^{N_{l-1}} w_{l,j,m}u_{l-1,m}\right] \tag{23}$$

$$= u_{l,j}(1-u_{l,j})\,u_{l-1,i}$$

with this (21) becomes

$$\frac{\partial J_p(W)}{\partial u_{l,j}} = \frac{\partial J_p(W)}{\partial u_{l,j}} u_{l,j}(1-u_{l,j})\,u_{l-1,i} \tag{24}$$

The term $\partial J_p(W)/\partial u_{l,j}$ represents the sensitivity of $J_p(W)$ to the output of node $u_{l,j}$. The node $u_{l,j}$ exhibits its influence on $J_p(W)$ through all of the nodes in the succeeding layers. Thus $\partial J_p(W)/\partial u_{l,j}$ can be expressed as a function of the sensitivities to nodes in the next higher layer as follows:

$$\frac{\partial J_p(W)}{\partial u_{l,j}} = \sum_{m=1}^{N_{l+1}} \frac{\partial J_p(W)}{\partial u_{l+1,m}} \frac{\partial u_{l+1,m}}{\partial u_{l,j}} \tag{25}$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial J_p(W)}{\partial u_{l+1,m}} \frac{\partial}{\partial u_{l,j}} \left[f\left(\sum_{q=1}^{N_l} w_{l+1,m,q} u_{l,q} \right) \right]$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial J_p(W)}{\partial u_{l+1,m}} f'\left(\sum_{q=1}^{N_l} w_{l+1,m,q} u_{l,q} \right) \frac{\partial}{\partial u_{l,j}} \left[\sum_{q=1}^{N_l} w_{l+1,m,q} u_{l,q} \right]$$

$$= \sum_{m=1}^{N_{l+1}} \frac{\partial J_p(W)}{\partial u_{l+1,m}} u_{l+1,m} (1 - u_{l+1,m}) w_{l+1,m,j}$$

This process can be continued for $\partial J_p(W) / \partial u_{l+1,m}$ and so on until we reach the output layer. That is the change in the weights in an internal node can be evaluated in terms of the changes in the layer higher up. Thus starting at the highest layer, the output layer, we evaluate the necessary change by (20), and propagate the "error" backward to lower layers. At the output layer we reach a boundary condition where the sensitiveness of the nodes in the last layer are derived from (19) as

$$\frac{\partial J_p(W)}{\partial u_{L,j}} = u_{L,j}(x_p) - d_j(x_p) \tag{26}$$

The learning rate μ, is a small positive constant, $0 < \mu < 1$, that determines how fast the learning takes place. True gradient descent requires that infinitesimal steps be taken. Since this is impractical the learning rate is introduced. The larger the learning rate, the larger the changes in the weights. If the learning rate is too big, that might lead to oscillations. This occurs because you are searching for minima, and if the steps are too big, you might pass a minimum, the direction of the gradient changes, and the next big step takes you back to the other side of the minima.

One way to retain rapid learning, and keeping the risk of oscillation low, is to introduce a momentum term to (20), that updates the weight, that is:

$$w_{l,j,i}(k+1) = w_{l,j,i}(k) - \mu \sum_{p}^{P} \left. \frac{\partial J(W)}{\partial w_{l,j,i}} \right|_{W(k)} + \alpha \left[w_{l,j,i}(k-1) - \mu \sum_{p}^{P} \left. \frac{\partial J(W)}{\partial w_{l,j,i}} \right|_{W(k-1)} \right] \tag{27}$$

where α is a small constant usually, $0 < \alpha < 1$, which determines the effect of past weight changes on the current direction of the movement in the weight space.

The learning law given by (27) is often referred to as the generalized delta rule. A further discussion on this learning rule can be found in *e. g. Rumelhart et. al, (1986)*, Chapter 8 or in *Hertz et. al. (1991)* chapter 6.

Note that the derivative of the sigmoid function given by (17) reaches its maximum for $f(y) = 0.5$ and, since $0 < f(y) < 1$, the derivative approaches its minima as $f(y)$ approaches zero or one. Since the change in weight is proportional to this quantity, it is clear that weights that are connected to units in their mid range are changed the most. In some sense these units are not certain wether to turn on or off. This feature is believed to contributes to the stability of learning procedure.

There exists no standard solution to how to choose the best momentum and learning rate. The optimal choice is totally dependent on the complexity of the problem. A usual approach is to start with large values, and decrease them until stabile learning is achieved. This approach usually gives you the fastest learning. It is common to let the ratio between the learning rates in successive layers be approximately 0.5. This to make the layers further down responsible for the big adjustments, and the layers higher up be responsible for fine adjustments. It is also common to change the learning rate and momentum during the training session, to start out with μ and α close to one, and decrease them as the algorithm gets closer to the solution. An example of such an approach can be found in *Jacobs, (1988)*

In the algorithm presented here the weights were updated every time a training pattern had passed through the network. Some prefer to accumulate the weight changes, and only do an adjustment after an epic of patterns have passed through. That is to update the weights episodically. The epic size can be the size of the training set, or smaller. No matter how often the weights are decided to be updated, it is important to present the training patterns in random order, because if the patterns are presented class by class, the updating might be very uneven. The gradient will point in one direction for one pattern, the weights adjust according to this, when the next class is presented, a totally new solution will be adapted to, and all that was learned before is likely to be lost.

If the number of patterns from each group differ strongly, it might be wise to present the patterns from the smaller class more often. This will speed up the learning, because if one class is very under represented, the network will soon adjust to the larger classes, but adjustments towards the smaller class will take place more rarely.

Even though suggestions on how to speed up the back propagation learning algorithm are flourishing in the literature, learning is slow. It has been proven by *Blum and Rivest, (1988)*, that in fact the learning process is NP-complete, *(Harel, 1987)*. However, their proof was done using a linear activation function, and it is still an open question if the proof can be extended to neural networks using nonlinear activation functions, such as the sigmoid function. NP-complete problems ar problems for witch all known solutions are essentially of the type "try all possibilities". With care, we can be a little more speedier than this, but it is generally believed that no algorithm to solve this type of problems can be substantially more efficient than the most obvious approach.

Another important question is how many nodes are needed in the hidden layers of a network. No one has yet come up with a general rule for this question. This largely depends on the complexity of the problem. The most common way of dealing with this question is to start with a small number of nodes, look at the performance, increase the number of nodes, look at the performance, and continue this procedure until there is little, or no improvement in the performance.

The last issue of debate I will mention is when to stop the training. Rarely a network converges to a perfect solution, either because there is none, or because the training takes too long. It is common to stop the training when the total mean square error between the output and the desired output has reached a low value, or when the training data has been presented to the network a specific number of times, or when there has been little or no change in the mean square error during the last training cycles.

4.6 The Output of the Network

It is easy to show that the output of a network, trained by minimizing the mean square error, is in fact an estimate of the least mean square estimate of the a posterior probability $P(\omega_i | \boldsymbol{x})$, that the class is ω_i, given that the pattern is $\boldsymbol{x}$.

Let's consider a network with one output node, (as applied in our experiment, see section 8), trained to solve a two class pattern recognition problem. The desired output of the network is given by

$$d(x) = \begin{pmatrix} C_1, \boldsymbol{x} \in \omega_1 \\ C_2, \boldsymbol{x} \in \omega_2 \end{pmatrix} \tag{28}$$

Typically $C_1 = 1$, and $C_0 = 0$. The mean square error given by (19) can be rewritten as the expected mean square error

$$J(w) = \frac{1}{2} E[(u(\boldsymbol{x}) - d(\boldsymbol{x}))]^2 = \frac{1}{2} E[u(\boldsymbol{x}) - E[d(\boldsymbol{x})]]^2 \tag{29}$$

$$E[d(\boldsymbol{x})] = C_1 P(\omega_1 | \boldsymbol{x}) + C_0 P(\omega_0 | \boldsymbol{x}) \tag{30}$$

Since $P(\omega_1 | x) = 1 - P(\omega_0 | x)$, we choose $C_1 = 1$ and $C_0 = 0$ so that (30) can be rewritten as

$$E[d(\boldsymbol{x})] = P(\omega_1 | \boldsymbol{x}) \tag{31}$$

Using this in (29) gives us

$$J(w) = \frac{1}{2} E[u(x) - P(\omega_1 | x)]^2 \tag{32}$$

Therefore by minimizing the mean square error between the actual and desired output, a least square estimate of the a posteriori probability $P(\omega_i | \boldsymbol{x})$, is given directly, by the output node. This result allows us to use the network as a Bayes classifier, and explains some of the success of neural networks used for solving pattern recognition problems.

5.0 Face Recognition

This chapter contains a review of, and references to some of the most important work done on face recognition. The first section contains a review of projects using traditional pattern recognition techniques. The second section contains a review of projects using neural network techniques to recognize human faces.

The problem of recognizing human faces has intrigued investigators for years. The first investigations on face recognition dealt with facial characteristics used by humans in recognizing individual faces. The work by a a French anthropometrist A. Bertillon (1853-1914), *(Encyclopedia Britanica, 1968)*, was perhaps the first formal attempt to make a face recognition system. He invented a system consisting of a concise verbal description that could generally be associated uniquely with a face. His system was used in France and many other countries for identification of criminals, until the use of fingerprints took over this task. His work formed the basis of how law enforcement agencies transmit descriptions of faces.

5.1 Face Recognition by Conventional Pattern Recognition Techniques

In modern days there have been many attempts to automate the face recognition procedures. Many studies have been conducted on how humans recognize faces, if specific features are more important, and how well these features can be quantized.

In the papers by *Goldstein et. al.,(1971, 1972),* they investigate how well a computer can identify a human face which is described by a person who is inspecting a photograph. They selected a set of 21 features with a range of legal values. An example of such a feature is eyebrows weight with values varying from one to five where one corresponds to thin, three to average, and five to thick.

Their data set consisted of 255 white male age 20 to 50, none had glasses or facial hair. A panel of 10 observers independently evaluated the 21 features for the entire data set. The average of the observers evaluation then became the official description of each feature vector for the data set.

In their experiment they got fifteen new subjects to view the photographs and interactively with a computer assign values to the feature vectors. They explored several algorithms for optimizing the man-machine system, to take advantage of the humans superiority in detecting noteworthy features, and the machines superiority in making decisions based on knowledge of the population and its statistics. Their results were encouraging and showed that if the subject described 10 or less features, and the machine filled in the rest, the population could be narrowed down to about 10 percent of the original size, and in 99 percent of the cases the target person was in this set.

Many other similar half automated feature based systems were investigated in the early seventies. The question naturally rose if it was possible to automatically extract features from photographs, and use these as a basis for a totally automatic face recognition system.

Profile silhouettes was a natural choice for many investigators. The profile trace can easily be obtained by thresholding a black and white photograph taken in high contrast.

In the work of *Kaufman and Breeding, (1976)* a set of correlation coefficients were extracted to serve as a feature vector. They chose the normalized circular autocorrelation function because of its scale, shift and rotation invariant nature. They also tested moment invariant feature vectors because they had some success with this earlier in recognizing airplanes. Their data set consisted of twelve silhouettes of each of the ten subject, taken with about a weeks interval. Choosing randomly six of the silhouettes of each person for training, and the remaining six for testing yield correct classification of up to 75 percent, for the autocorrelation vector. When they increased the training set to nine, and used the remaining three as test set they got up to 90 percent correct answers. Although these results were encouraging, the results varied a lot, depending on what pictures were in the training set and which pictures were in the testing set. Their results for the moment invariant feature vector were not as good, the best performance was then around 60 percent.

Harmon et. al. (1981), also used profile silhouettes for automatic face recognition. From the profile trace, eleven automatically placed fiducial marks were located. From these marks a total of thirty-eight geometric measurements were calculated. These were typically angles, areas, and distances. Out of the thirty eight features tested, seventeen were retained for the classification task.

Their database consisted of 130 people photographed four times at weekly intervals. Twenty of the subjects were deleted from the data base because of distorted data or badly placed fiducial marks. The photographs were scanned and the profile trace were extracted. The average value of three of the vectors were used for training or to construct prototypes of each person, and the fourth one was used for the classification task. None of the subjects had facial hair, or wore glasses.

Harmon et. al. (1981), conducted several different experiments with this data set. Testing one person against one average vector, a positive match was defined if the difference between the vector and the average vector was less than a certain threshold. The result of this was 96% successful. When the vector was matched with all the vectors, the match was correct in 96% of the trials, and in the remaining 4% the correct average vector was in second place. They also tested the possibility of rejecting strangers, or unknown to the data base, and found that a difference threshold could be set so low that all strangers could be rejected. When the classification task was performed on nine features located from the nose upwards the accuracy was about 85% correct.

The approach by *Takahashi, et al. (1990)*, is slightly different. They use isodensity lines of the human face for recognition. The isodensity lines are the boundaries of contrast gray level areas after quantizing an image. They can best be compared to the height curves in a topographical map. The isodensity lines reflect the relief of the face, and therefore take into account the three dimensional nature of the face. Identification was done by comparing the isodensity lines of different people. The authors argue that this type of features give a stable description of the whole face, over a range of normal face positions.

The data base consisted of 18 different people, among these some with glasses or a short beard growth. Some of the subjects changed their hair-do and makeup slightly between the pictures. Each subject was photographed twice. The isodensity lines were extracted by using a Sobel filter and thresholding. The experiment consisted of matching two and two pictures of the same person and of different people. They went through two rounds of matching, first a global matching which compared the projections of constant gray level areas in horizontal and vertical directions. This matching got 16 of the 18 pair of "same person" and 283 of the 306 pairs of "different people". The next stage was a fine matching where they compared characteristics of the fine isodensity lines. After this matching they got a 100% matching in both the same person and the different person category.

5.2 Face Recognition Using Neural Networks

Kohonen et al, (1981), were probably the first to do face recognition using whole images of faces, represented as image vectors consisting of intensity levels. They used the whole face images, without any extensive form of preprocessing. This also meant that things like glasses and facial hair didn't interfere with the classification task.

In their 1981 paper they invoked the task of face recognition to demonstrate that optimal associative mapping can be used for pattern recognition. The associative network is a neural network with essentially a one layer network designed to map the input vectors into a set of "memorized" vectors stored in the network. The data set consisted of 10 different people photographed from five different angles. By training and testing on different pictures of each person they demonstrated that an associative mapping is able to yield a correct classification of a person, even when presented a picture of a person viewed from a different angle than the ones stored in the memory.

Kohonen also used faces in a demonstration of auto associative recall, a network which is presented with a vector, and then outputs the stored vector in the network which it associates with the input vector. In the demonstration they memorized 100 pictures of 100 different people and demonstrated that the auto associative mapping is able to reconstruct the original images when presented with a distorted version of it. This version could be the image added noise, or other picture missing some parts. In the book by *Kohonen, (1984)* he presents results where 100 pictures were stored in the memory, and showed how they could be recognized and reconstructed when the memory was presented with between 10% and 50% of the images.

In the paper by *Perry and Carney, (1990)*, they investigate using a multilayer perceptron neural network with the back propagation learning algorithm for human face recognition. Their data base consisted of six pictures taken of three subjects. The images were of size 482 X 499, and the gray levels were stretched to the range from 0 to 127. From the original six images the investigators composed a data set of seventy two images by applying various transformations to each image to simulate translation, rotation, and perspective changes. They also generated noisy images and images with missing data. The data set was then compressed to 32 X 32 by averaging over 16 X 16 windows. They randomly selected twelve of the images for a testing set, and used the remaining sixty as the training set.

They trained several networks with different learning rates, and different numbers of nodes in the hidden layer. They varied the learning rate form 0.1 to 2.1, and the number of hidden nodes from 4 to 24. They trained the networks until it met a 0.01 RMS convergence criterion, or until it had been presented with the images 500 times. Each person was trained as one class. They found 100% recognition rate when they used eight or more nodes in the hidden layer, and they defined correct classification as if the output node corresponding to the target class had the highest output. If they used a stricter classification criterion, that the output of the target class node must be above 0.8, and the output of the other nodes must be below 0.2, they needed 16 or more nodes in the hidden layer. They compared their results to the K-nn classifier and found that also this one gave 100% correct classification if the one nearest neighbor was used for classification. If they used more than one neighbor, the classification was not always correct.

In a preliminary study for this project *Castain, (1990)*, demonstrated that backpropagation neural networks with 20 nodes in the hidden layer could recognize five different people with 100%. The data set consisted of five different subjects photographed nine times. The original images were 512 X 512. He trained one network using the full images, and two networks using images compressed to 32 x 32 by averaging. He used four images of each person in the training set, and the remaining five images in the test set.

6.0 Data and Preprocessing

This chapter contains a description of the data used for our face recognition project. It gives a description of the data collection and the demographics of the data base. This to illustrate the variety of people that were included in the survey. Section 6.3 contains a description of the compression method we used for preprocessing of the images. Last section 6.4 contains a discussion on why wee selected this compression method. It also demonstrates what information we loose by using this technique.

6.1 The Data

Our database consists of photographs of 511 different people. We took approximately 20 pictures of each person. The total number of pictures we collected was 10221. The data collection was done during a five month span.

The subjects were mostly laboratory employees and visitors to the laboratory. The subjects were recruited, and informed about the project by flyers distributed around the laboratory, flyers posted in strategic places, and advertisements placed in the laboratory news bulletin.

The pictures were taken using a video camera. We used the software VideoPix from SUN to freeze and store the digitized images. All the photographs were taken under bright lighting conditions. We used two photographic lights to illuminate the subjects, and to minimize face and background shadows. We used a piece of blue cardboard as background in all the pictures.

The subjects were sitting in a comfortable chair, facing the camera. An illustration of the set up is provided in Figure 7. The subjects were encouraged to relax, talk and feel comfortable during the three minutes long photo session. The pictures we took attempted to catch a variety of normal facial expressions. All the subjects were asked to tilt their heads up, down, and to the side for some of the pictures. Some of the subjects were asked to turn their head slowly from side to side, so that we could take pictures from all different angles. The subjects wearing glasses were asked to remove their glasses in some of the pictures. Subjects with long hair, were asked to pull their hair to the front in some pictures and to the back in others. Some of the male subjects were kind enough to shave or grow their beard for us, so that we could have pictures of them during these changes. Some subjects were asked to come back for a second photo session, some days or weeks after they had been photographed the first time. Some of the pictures contained the subjects shoulders, and others not. We were not strict about the exact position of the subject in the image. Most of the subjects changed positions several times during the photo session. Figure 8 shows two picture sequences.

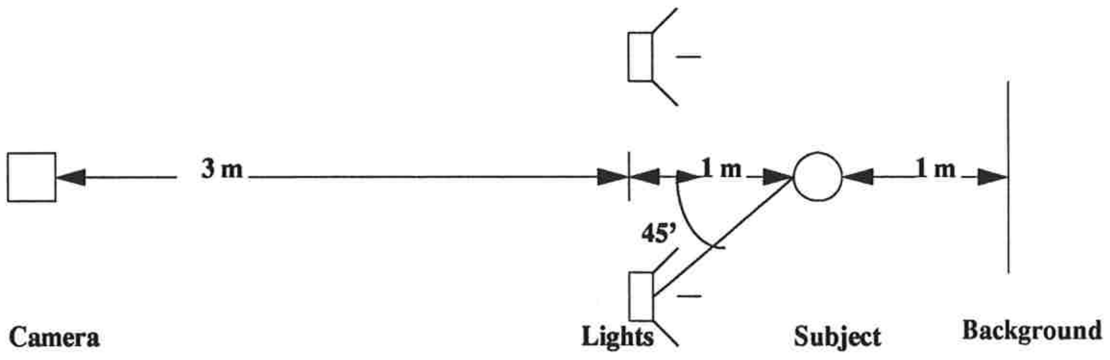


FIGURE 7. The data collection set up

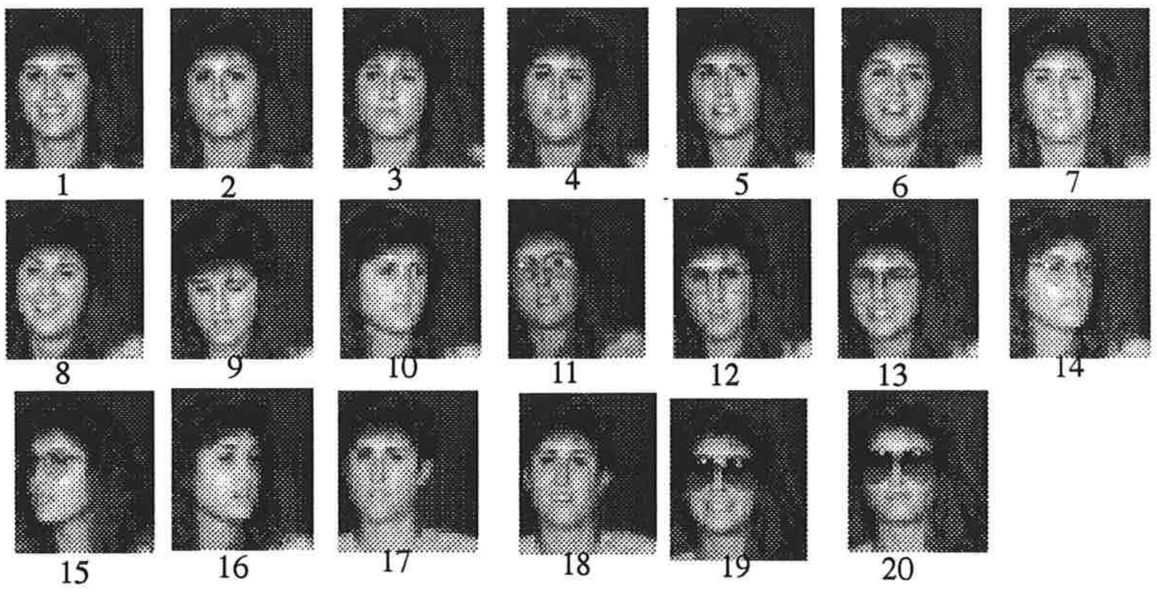


FIGURE 8. a) Sequens of pictures

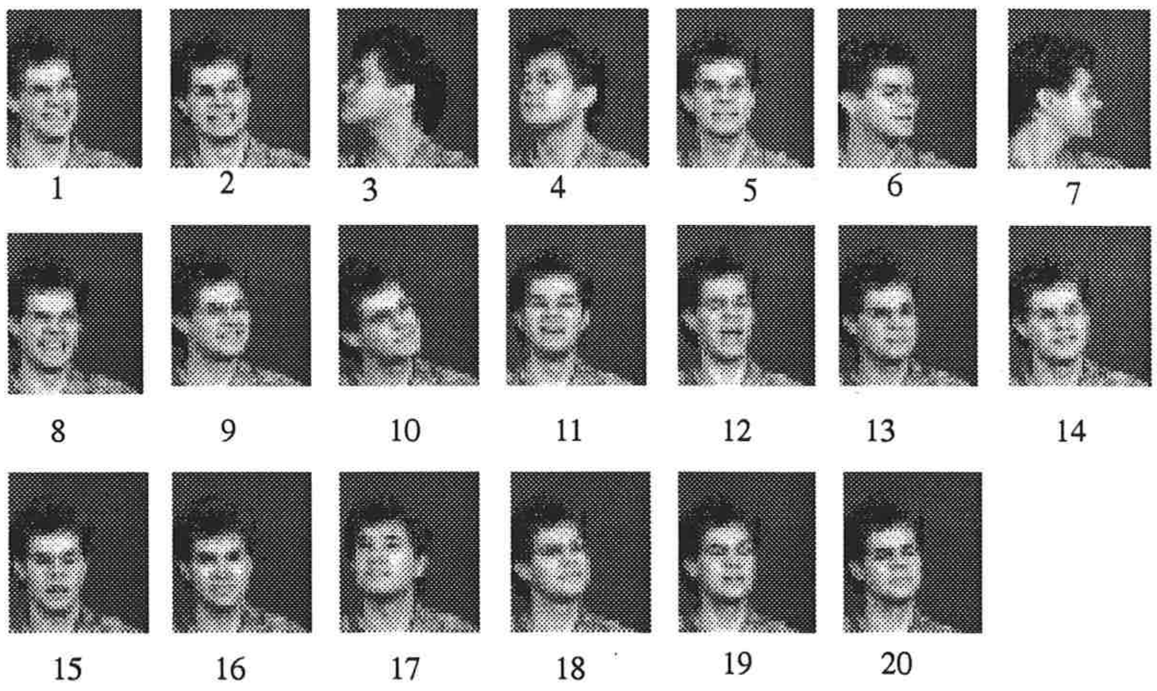


FIGURE 8. b) Sequens of pictures

6.2 Demographics.

To get an idea of what kind of population our results are based on we did a demographic survey of the people we took pictures of. A form was completed by the project team at the same time as we took the pictures. The demographic features we made notes about are included in table 2.

Hair color and beard color were allowed to have more than one value. We also wrote down comments if there was anything special to note about the subject. This could be things like change in hair style, or if the subject was wearing a hat or dark glasses in some of the pictures. In table 2 we have summarized the statistics for the demographics of the data base. A total of 511 subjects make the basis for the values in the table.

TABLE 2. Demographics of the 511 subjects in the database.

Feature	Value1	Value2	Value3	Value4	Value5
Sex	male	female			
% of population	60%	40%			
Skin Tone	light	medium	dark		
% of pop	75%	25%	0%		
Hair color	blond	brown	black	grey	red
% of pop	14%	77%	19%	34%	5%
Hair length	short	medium	long		
% of pop	72%	18%	10%		

TABLE 2. Demographics of the 511 subjects in the database.

Feature	Value1	Value2	Value3	Value4	Value5
Hair tone	light	medium	dark		
% of pop	31%	63%	9%	5%	
Hair loss	no loss	light loss	medium loss	pronounced	
% of pop	70%	16%	9%	5%	
Beard	no beard	beard	mustache	both	
% of pop	77%	0%	13%	10%	
Beard color	blond	brown	black	grey	red
% of pop	2%	20%	4%	28%	2%
Glasses	yes	no			
% of pop	34%	66%			
Age group	0 - 15	16 - 25	26 - 35	36 - 55	56 +
% of pop	0%	13%	26%	59%	2%

6.3 Preprocessing

The original images were of size 640 x 480 X 7 bits. Because much of the images consisted of background, we cropped most of the background by using the following algorithm.

- Estimate the intensity of the background.
- Calculate the center of mass, subtracting the average background value from each pixel.
- Crop the image by deleting everything outside a 350 X 400 window, centered around the center of mass.

This simple method worked well on the majority of the images. If the subject was situated close to the edge of the image, this method tended to cut off some of the side of the face of the subject. If the subject was wearing a white shirt, had dark hair, and the shoulders were present in the picture, this would cause the center off mass to be pulled to low, and the top of the head of the subject would be cropped off.

The grey levels in the images were stretched to an 8 bit range, to enhance the contrast in the images. The photographs were then compressed to a size of 35 X 40. The compression was done by letting each pixel in the compressed image represent the average value of the pixels in a 10 X 10 window in the original cropped image. Figure 9 shows a sequence of an original, stretched, cropped and compressed image.

The image was then scaled from -1 to +1, and fed into the backpropagation neural network as an image vector.

A copy of the compression and pre-processing program code written by R. Castain is included in Appendix A

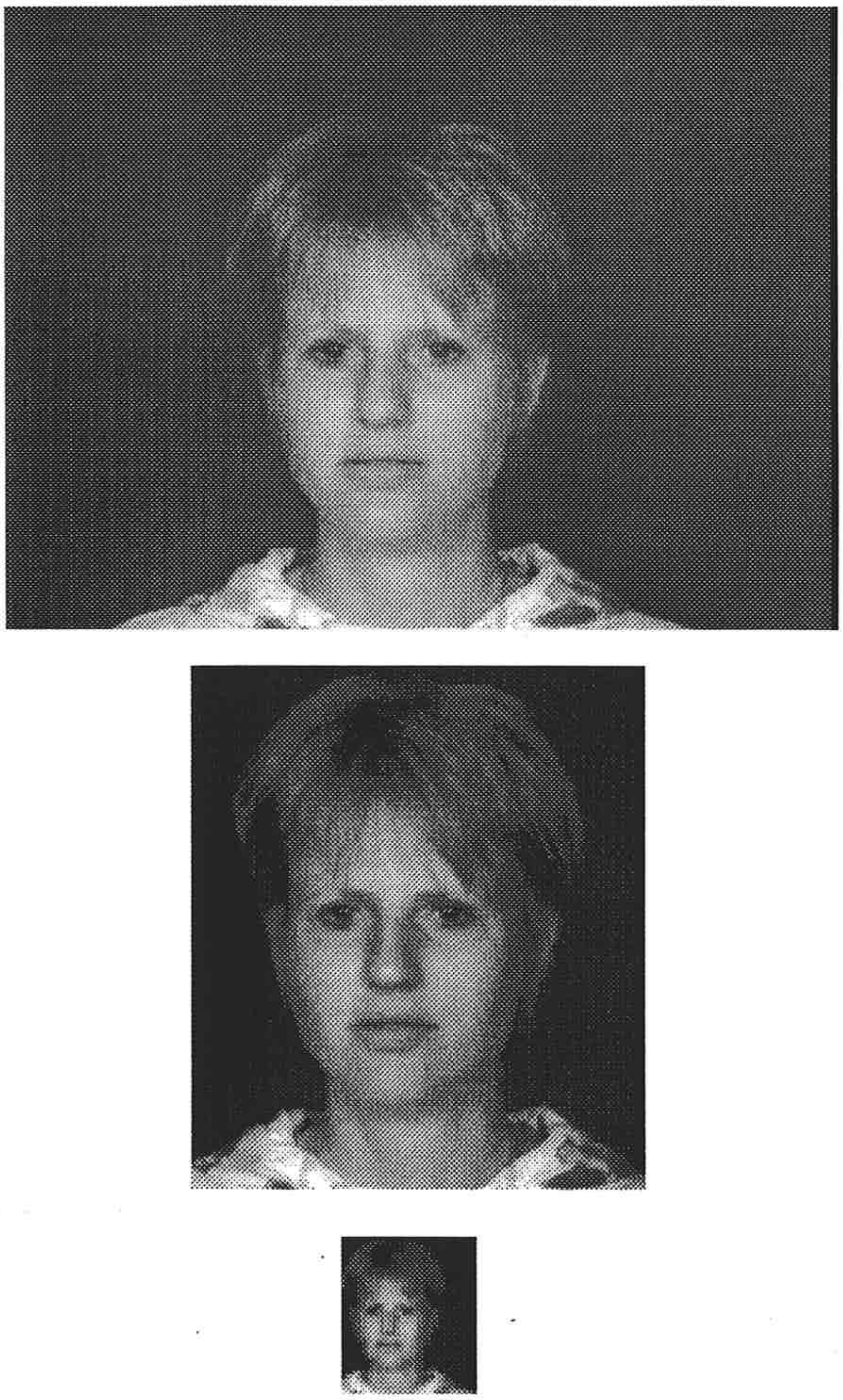


FIGURE 9. Sequens of original, stretched and cropped image, and compressed images, The original and cropped images are scaled down by a factor of 50%, and the cropped image is scaled up by a factor of 50%.

6.4 Justification of Preprocessing

We cropped the images because we wanted to make sure that the classification was based on the faces of the subjects, and not on shadows or other noise that occurred in the background of the photographs. We also did it to reduce the size of the pictures to save training and testing time.

We wanted to be able to train and test networks using all of our 10 221 images, and we wanted to train as many networks as possible, so that we could test different scenarios and training schemes. Training on image vectors of size 307 200, would take far more time than we were interested in spending. By compressing the images by 100, we could easily train five networks a day, on each of the three SUN SPARC stations we had available for the project.

We looked into several other compression techniques that would retain all or most of the high frequency information. The technique used by *Bradley and Brislawn, (1991)* is a subband coding technique. In Subband coding, *(Rabbani and Jones, 1991),* an image is first filtered to create a set of images, each of which contains a limited range of spatial frequencies. These images are the subbands. Since each subband has a reduced bandwidth compared to the original full band image, they may be down sampled. However ideal filters are unrealizable, and it is necessary to use filters with overlapping response in order to prevent frequency gaps in the images represented by the subbands. The problem with overlapping filters, is that aliasing is introduced when the subbands are down-sampled. To overcome the aliasing problem Bradley and Brislawn used a quadrature mirror filter in the form of a wavelet filter, *(Mallat, 1989)*. This filter allow for an aliasing free reconstruction of the downs-sampled images. In the work by *Bradley and Brislawn, (1991)* they split the original image into four subbands, low-pass - low-pass, high-pass - low-pass, low-pass - high-pass, and high-pass - high-pass. They then replaced the low-pass - low-pass band by splitting this image into four new subbands. They continue this process until the desired compression has been reached. (Figure 10 shows an example of this technique). To get an even higher compression rate they combined this technique with a vector-quantization technique. In image quantization the original image is first decomposed into *n*-dimensional image vectors. These vectors can for example be formed by letting an $m = l X n$ block of pixel values be ordered into an n dimensional vector. Each of the image vectors is then compared with an collection of representative templates or code vectors, taken from a previously generated code book. Details of how this codebook was generated can be found in *Loyd, (1982)*.

We considered using this technique, by letting the network train on the final low-pass - low-pass subband, and certain areas of the other subbands. We also considered extracting the eye - nose regions from the other subbands and include this in the image vector formed by the low-pass - low-pass subband. We did however not try this, but for future investigations it could be interesting to see if this extra information would improve the performance of the networks.

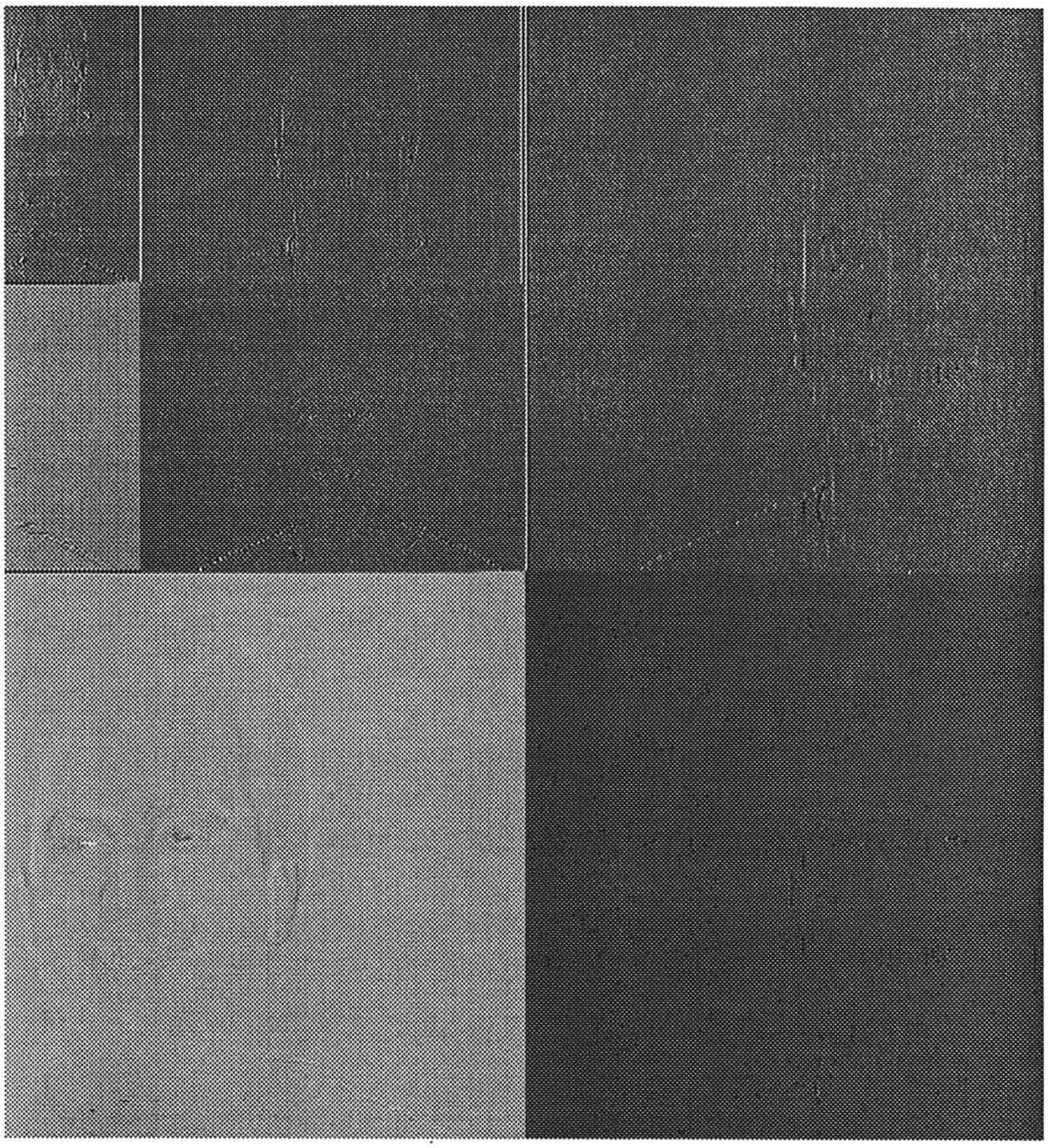


FIGURE 10. Wavelet compressed image.

For similar reasons we looked into using the discrete cosine transform. By using the cosine transform the image is also separated into different subbands. Again we considered using the DC band, and some areas of some of the other frequency bands, to add more information to the network. (Figure 11 shows an example of a cosine compressed image.)

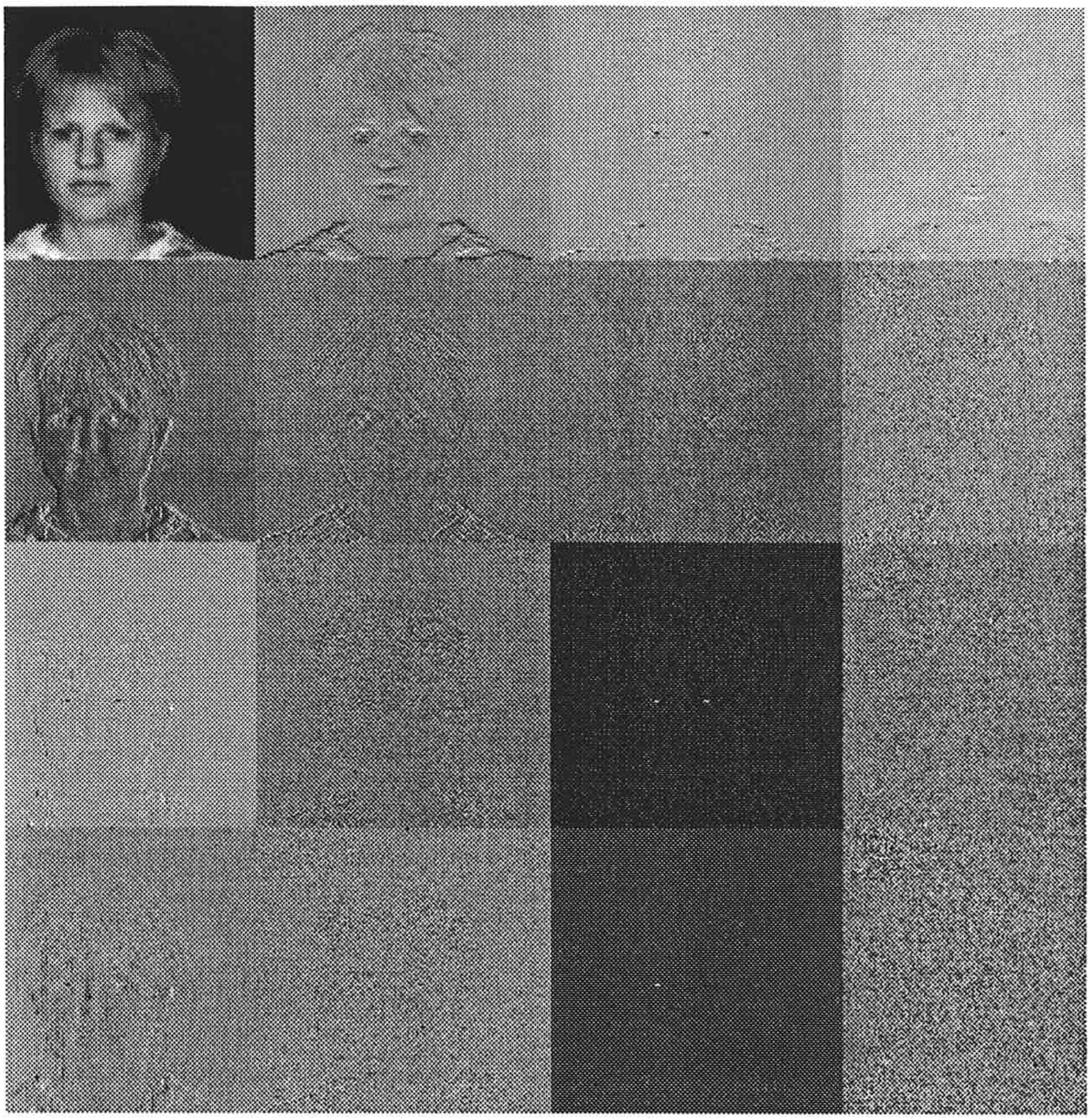
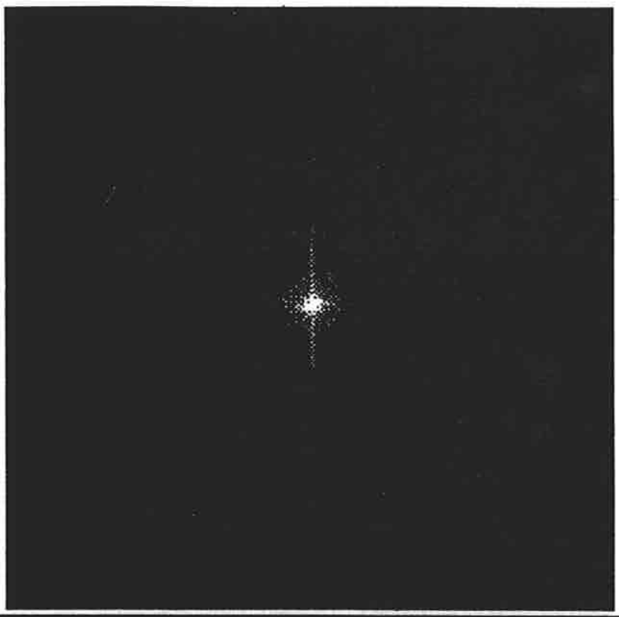


FIGURE 11. Cosine compressed images

By examining the 2 dimensional power spectrum of a facial image, and the histogram of the power spectrum (figure 12, original, power spectrum and histogram of the power spectrum) we see that the low-pass components are the dominating frequencies in the image. This statement is also verified by looking at the various frequency bands in figure 10 and 11, because of this, we chose to use a compression method that retain the low-pass frequency components and discharge the higher frequency components. A trivial and safe way of doing this is simply by averaging.



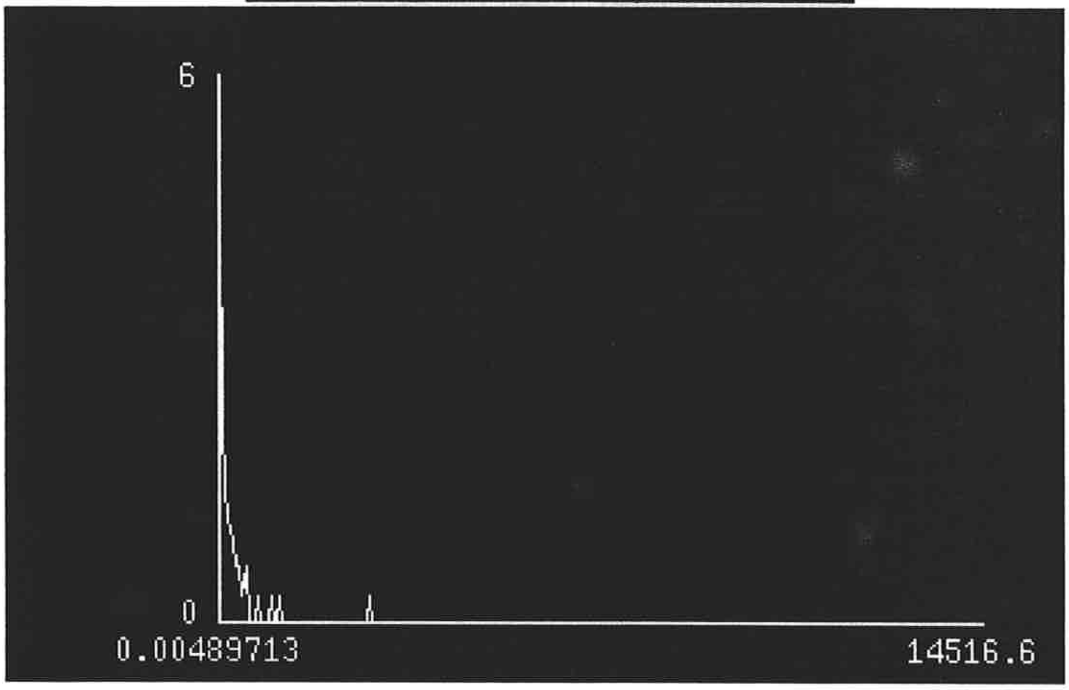


FIGURE 12. Power spectrum and histogram of power spectrum, of the original image.

An interesting study done by *O'Toole, et al. (1988)*, supports our choice of compression technique. They did a survey of how well human subjects, and a neural network could recognize spatially transformed faces. They found that the transfer between normal and low-pass filtered faces was good. The transfer between normal and high-pass was not as good, but improved by how familiar the face was. Last, the transform between high-pass and low-pass was poor. 42 photographs of male students, and 42 photographs of female students were used as stimuli, and sixty students were used as the human subjects to learn and recognize the different pictures. The neural network architecture they used was a partially connected Kohonen inspired network which they designed especially for this experiment.

7.0 Preliminary studies of data, training and network architecture

In this chapter I will summarize the findings of some preliminary training of 20 networks. A set of networks trained with the goal of finding a usable network architecture and parameters for the Backpropagation learning law. We also used the networks to decide on the specific way we wanted to compose the training sets.

7.1 Composition of the Training Sets

Our database consists of approximately twenty pictures of each of the subjects. We randomly selected ten pictures of the target subject, and five pictures of each of the other subjects in the training set. All of the pictures of every subject was included in the test file.

Before we decided on this composition of the training and testing files, we had done some preliminary training of approximately twenty networks where one of the goals was to find the best way to compose the training sets. We came to the following conclusions:

- It is important to select the pictures used for training randomly, to ensure an evenly distributed amount of pictures with different postures and backgrounds.

In some of the networks we trained, we had used the five first pictures of each of the non target subjects and the ten first pictures of the target subjects. In theses pictures the subjects were mainly looking straight ahead. We naturally found that the generalization of the network to unseen postures was bad. We then randomly selected the pictures of the target subject, and found that the generalization was better. The training file now consisted of pictures of the target where the target subject was looking straight ahead, some profile pictures, and some looking up or down. What we found in testing these networks was that confusion (wrong classification) of pictures took place if the subject was photographed in the same postures as the target person. For instance, the network would associate profile images with the target, if the only profile images in the training set was of the target. This problem disappeared when we randomly selected all the pictures in the training file.

The picture taking occurred in several different rooms with different lighting conditions, and different distances from the camera to the subject. Even though we used the same photographic lights and background, the different reflection conditions in the different rooms, did have an effect on the images. In fact, if the target person was the only person in the training set, whose pictures were taken in a specific room, the network would train to classify of the background and not the face of the target person. During testing we would see a large amount of confusion that could only be explained by the fact that the subjects had been photographed in the same room, and that the classification was based on the background. Including one or more subjects whose pictures were taken in the same room, eliminated this problem. A similar effect could sometimes be found in small training sets, where the network did the classification of the target, based on the shirt or shoulders of the target. We concluded:

- It is important to include other subjects in the test set from the same group as the target.

If the target was the only person from its group, the network would confuse a large amount of the subjects from the targets group in the testing set.

In these preliminary studies we used 5, 20 or 100 subjects in the training file, and tested against a file of 200 subjects. Therefore at least 50% of the subjects in the testing file were unseen by the network. During testing, we found that all the confusion occurred amongst these unseen subjects, with the exception of the confusion which occurred in connection with postures. This was true when we included five or ten images, in the training set, of each of the subjects in the non target group. When we included only one image of each of the non target subjects, some confusion also occurred amongst the subjects whom the network had seen before. The confusion in the unseen group also increased. We concluded:

- The network primarily confuse pictures of unseen persons, or pictures of postures.

There was a low false rejection rate of target subject, in all of the networks, except when it came to the case of extreme, previously unseen angles. The recognition improved when we increased the number of target pictures from five to ten. We did not want to use more than ten target pictures in the training file, because we felt it was important to have a variety of unseen pictures of the target in the testing file. This was done to determine if the network could generalize well. We rejected the option to include all but a few images in the training file, and then train many networks, with different combinations of the target pictures included, to find out how well we could train the network. We concluded:

- Including 10 pictures of the target person in the training set, and 10 unseen pictures for the testing file would give a realistic impression of the performance of the network.

7.2 The Architecture of the Neural Network

We applied the back propagation algorithm for the classification task. We experimented using different numbers of nodes in the hidden layer. We found that improvements in recognition and less confusion occurred when increasing the number of nodes from 10 to 15, and from 15 to 20. Increasing the number of nodes from 20 to 30 did not give any significant improvement of results, but the training time did (of course) increase. We therefore decided on using 20 nodes in the hidden layer.

We also experimented with different compression rates. We compressed the original 460 X 640 images down to a set of images size 70 X 80, and to a set of images size 35 X 40, The networks trained with the 70 X 80 nodes gave slightly better results, but the improvement in the results was not enough to justify longer training time, we therefore decided to use the higher compression. This gave us 1400 input nodes.

Our findings is supported by the face recognition done by P*erry and Carney, (1990)*, they used 1032 input nodes and found that they needed 16 or more nodes in the hidden layer to get 100% correct classification. Also in the preliminary studies done by *Castain, (1990)*, he found that 20 nodes in the hidden layer yield good classification results.

We experimented with how long it was necessary to train the networks. We tested a variety of networks after they had trained for 500 to 60 000 cycles or presentationes of pictures.

We found that most of the change occurred during the first 1000 presentations of pictures. After this, only minor adjustments, that had little or no effect on the classification occurred.

8.0 The Face Recognition Experiment

In the following section I will report on the architecture and results of the face recognition experiment. The details of the results are summarize in tables included in appendix B.

In section 8.1 I report on the parameters we used in the Backpropagation algorithm. In section 8.2 I explain how we constructed eight different training sets, and what our motivation for choosing these specific training scenarios was. In section 8.3 I introduce the criteria we used for classifying errors. In this section we separate the errors into two groups, not classifying the target correctly, and not classifying the other subjects correctly. In the four subsections of section 8.3 we analyses these two error types, to search for differences in performance between the different training scenarios. We then grouped the pictures that caused errors, to see if we could find any patterns between these. In section 8.4 I rewed the performance of five personnel verification systems, and compared their performance with the performance we achieved in the face recognition experiment.

8.1 The Architecture

In this section I will specify the parameters used in the backpropagation algorithm. The motivation for each choice of parametric values, can be found in the review of the algorithm, section 4.5, or in Chapter 7, witch contains a review of preliminary trained networks.

We used the back propagation algorithm in the NeuralWorks Professional II Plus software package from NeuralWare Inc., run on a SUN SPARC station IPC.

The network had the following parameters:

Input nodes: 1400

Hidden layer nodes: 20

Output nodes: 1

The bias weight was fully connected to the hidden layer, and to the output layer. The output layer was fully connected to the hidden layer, and the hidden layer was fully connected to the input layer. All weights were initially set to random values between -0.1 and +0.1, this because if all the weights start out with equal values, and the solution requires that the network develope unequal weights, the system can never converge to a stabile solution. The reason for this is that the error is propagated back through the network in proportion to the value of the weights. This means that all the hidden units will get identical error signals, hence all the weights will make the same update, and the system will not learn, and adapt to a solution.

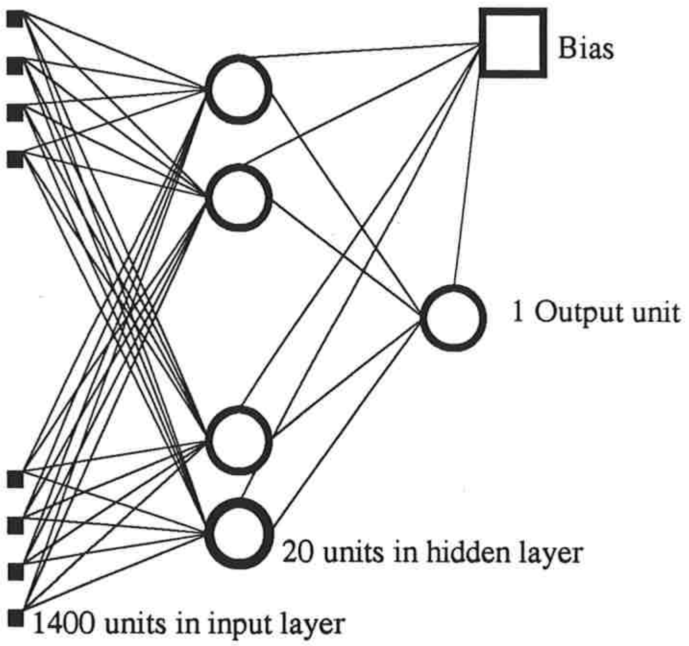


FIGURE 13. The backpropagation neural network architecture we used

Both the hidden and output nodes used the sigmoid transfer function given by (14). The input nodes or terminals served only as a feeders of patterns to the hidden layer. The input vectors were scaled to the range from -1 to +1, to keep the network from saturating which prevents learning. We used the following parameters for the delta learning rule given by (27):

Learning rate, hidden layer: $\mu_1 = 0.3$

Learning rate output layer: $\mu_2 = 0.15$

Momentum: $\alpha = 0.4$

Gain: $g = 1$

In the preliminary training of networks, we trained some networks with higher learning rates and momentum. This worked nicely in some of the networks, but not for all. We therefore decided to lower the learning rates and momentum to a level where all the networks trained satisfactory. Section 4.5 has a discussion of the parameters.

The network was trained to give an output of 1 for the target person, and 0 for the other subjects. One of the features of the sigmoid transfer function is that it can never reach its extreme values of 0 and 1, therefore the actual output after training was for the target class $d(x_1) > 0.9$, and for the other class $d(x_2) < 0.1$.

8.2 The Training Sets

In this section I will explain how we constructed eight different training sets, and how we hoped these would enlighten our questions on how to achieve efficient decision boundaries, and of the question if the training sets needs to reflect all the people in the data base, or if a carefully selected combination of people can assure the same decision boundaries.

Based on the demographic information we divided all the subjects into groups according to the subjects sex, skin tone, hair tone and hair color. These are natural clusters to choose, since the classification is performed based on grey level intensities. We then randomly selected three or four subjects from the eleven most populous groups. For each of the 37 selected subjects we trained four networks that we tested against the 9461 pictures of 477 persons in our data base at that time. Subsequently, we decided to train four more networks for 11 of the subjects, these networks were tested against 10221 pictures of 511 people. The eight networks were as follows:

- The target subject versus members from its own demographic group, the training set consisted of approximately 2% of the pictures in the data base. This training scenario is called **VsGroup** in what follows.
- The target person versus pictures of two or three subjects from each of the other groups, this training set also included two or three subjects from the target's demographic group. These training sets consisted of approximately 4% of the pictures in the database. This training scenario is called **VsSome** in what follows.
- The target person versus a selection of people from each group, and most of the members of the target subjects group. These training sets was constructed for approximately 5%, 10% and 15% of the data we collected. This training scenario is called **VsGr-Some5, VsGr-Some10, VsGrSome15**, respectively, in what follows.
- The target subject versus a random selection of pictures of the entire population in the database, this was done selecting approximately 5%, 10% and 20% of the pictures in the database. This training scenario is called **Vs5%, Vs10%, Vs20%**, respectively in what follows.

The training files for the subject versus members of its own group consisted of pictures of between 20 and 30 subjects. The number dependent on the number of people in each group. From the largest groups we used approximately 1/3 of the subjects, and from the smallest groups we used all, except for the target subject, to form the training sets. The training set of two or three persons from each group consisted of 70 subjects. These included members from all the groups consisting of more than one member. The above described training sets consisted of five randomly selected pictures of each of the persons in the training set, and ten randomly selected pictures of the target person. For each target person the same set of target pictures where used. In the three random networks, each picture in the data base had an equal probability of 5%, 10% or 20% of being selected for the training set. These sets therefore consisted of pictures of a much larger number of different people than the other training sets.We forced the selection of the target pictures to assure that ten pictures of the target was selected. The set of target pictures were therefore different in each of the random training sets. In the first round of network training we trained the

VsGr, VsSome, VsGrSome5 and Vs20% scenarios. In our second round of training we trained the VsGrSome10, VsgrSome15, Vs5% and Vs10% scenarios.

By constructing these four different types of training scenarios. We wanted to test what sort of training set is necessary to get adequate decision boundaries. Boundaries that would give a correct classification or verification of the testing set, the testing set consists of all the pictures in the data base. By the construction of training sets we could test the following:

- If only using pictures of people looking similar to the target, thus only using people from the targets demographic group would be adequate.
- If using a small selection of people from all the other democratic groups would be adequate
- If using a small selected set of non similar and similar people would make good decision regions, and if they do, how many is necessary?
- If using pictures of all the people in the database would be the only way to make decision boundaries, that classified all the pictures in the data base correctly, and if so, how manny pictures are needed.

The exact number of pictures in each training file can be found in table 14 in appendix B

TABLE 3. Demographics of the 37 target subjects.

Feature	Value1	Value2	Value3	Value4	Value5
Sex	male	female			
% of targets	54%	46%			
Skin Tone	light	medium	dark		
% of targets	65%	35%	0%		
Hair color	blond	brown	black	grey	red
% of targets	19%	70%	27%	24%	5%
Hair length	short	medium	long		
% of targets	65%	22%	14%		
Hair tone	light	medium	dark		
% of targets	38%	62%	0%		
Hair loss	no loss	light loss	medium loss	pronounced	
% of targets	76%	16%	0%	8%	
Beard	no beard	beard	mustache	both	
% of targets	89%	0%	3%	8%	
Beard color	blond	brown	black	grey	red
% of targets	3%	5%	3%	3%	3%
Glasses	yes	no			
% of targets	43%	57%			
Age group	0 - 15	16 - 25	26 - 35	36 - 55	56 +
% of targets	0%	19%	32%	49%	

Since the number of target pictures in each training file was severely under-represented, ranging from less than 1% to 10% at the best, we added replicas of the target persons images, to guarantee that the number of target pictures would be at least 10% of the total number of pictures in the training sets.

We stopped training after a specific number of iterations (se section 4.5 and section 7.2). For the smaller training files consisting of 1 000 or less pictures, we stopped the training after 10 000 cycles, implying that each non target picture was presented at least 10 times to the network, and each target picture was presented at least 100 times to the network. For the large training files, consisting of up to 2 500 pictures we stopped the training after 15 000 cycles, implying that each non target picture was presented at least 6 times, and each target picture was presented at least 60 times to the network. Most of the adjustments took place in the first 1/3 of the training. This can be seen from Figure 14 that show a typical plot of how the total mean square error behaves during training. The training time was approximately 1 hour for the small training files (1000 pictures or less), and 1.5 hours for the larger training file. The testing of 10211 pictures took about 2.5 hours on a SUN SPARC Station IPC.

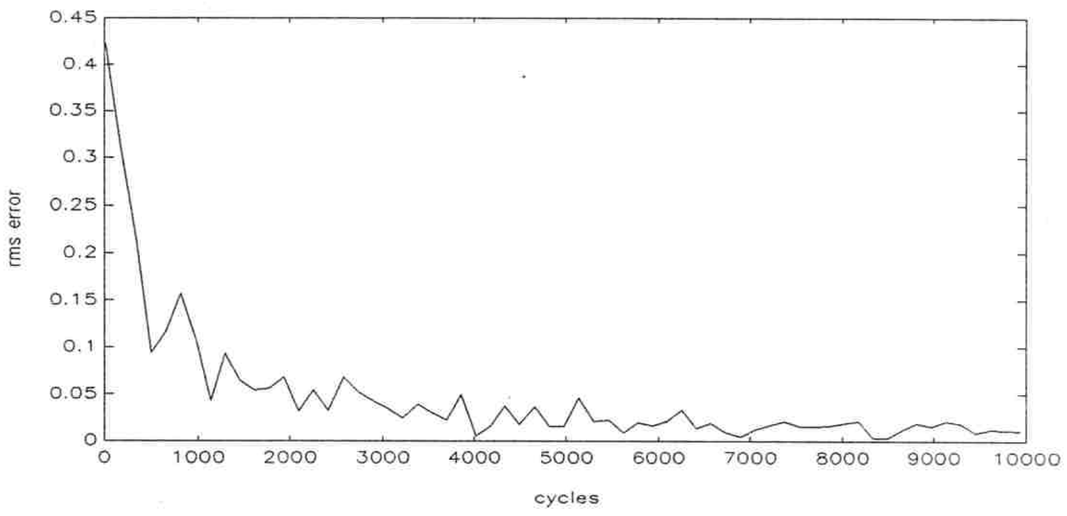


FIGURE 14. RMS Error, as seen during training.

8.3 Error Classification

In this section I will introduce the way we chose to classify errors. We separated the errors into two classes, false rejects and false accepts. These errors correspond to not classifying the target correctly, and not to classify the other subjects correctly. We did a binary and trinery division of the outputs of the networks, to classify wether the output was correct or not.

We have divided the errors into two classes, **false accepts** and **false rejects.** False accepts refers to classification of a non target person as the target. False rejects refers to not classifying a target as the target.

For each of the networks, we did two types of error classification. First we did a binary division, we will refer to this as **binary** errors, using the following classification rule:

$$x_p \in \omega_0, d(x_p) \le 0.5 \qquad (33)$$

$$x_p \in \omega_1, d(x_p) > 0.5$$

where $d(x_p)$ is the output of the network, and ω_1 is the target class, and ω_0 is the other class. This is a division where we choose to classify the target as the class with the highest a posterior probability (see section 4.6), where $P(\omega_1 | x_p) = d(x_p)$ and $P(\omega_0 | x_p) = 1 - d(x_p)$.

The second division we made was a trinery division. We will refer to it as **trinery** errors, and use the following definition:

$$x_p \in \omega_0, d(x_p) < 0.2 \qquad (34)$$

$$x_p \in \omega_1, d(x_p) > 0.8$$

$$x_p \in \omega_3, 0.2 \le d(x_p) \le 0.8$$

where ω_3 is a "don't know class". In this division we have included a region where we say that if the output is in this region, it is undetermined who the person is. In a personal verification system this indicates that a new picture needs to be taken in order to accept or reject the person. In both error classifications we say that it is a false acceptance if x_p is classified as ω_1, when it belongs to ω_0, and a false reject if x_p is classified to ω_0 when it belongs to ω_1.

The difference in the results between the trinery error and the binary error was that a large number of the false accepts in the binary case were moved to the don't know class in the trinery case. The false rejects from the binary case remained false rejects, or were moved to the don't know class. Since the aim of this project is a security system, the detailed analysis of the types of errors will be conducted on the errors in the trinery division.

8.4 Error Analysis

In this section we will do an error analysis to investigate if there are any differences in performance between the different training scenarios. We then investigate what pictures caused the errors. This to find the limitations and performance of the system.

8.4.1 False Rejects

As described in chapter 6, the pictures of each person included profile pictures, slight sideways pictures, up- down- pictures and a variety of different facial expressions. In this section we will investigate how many of these pictures the network verified correctly, and if the different training scenarios performed differently. We choose to use a non parametric statistical test for this inference, because we suspected a serious violation of the assumption of a normal distribution, which form the basis for most parametric inferences.

Table 4 gives a summary of the average performance of each of the training scenarios. The mean percentage is calculated based on the average number of false rejects, and the Std is the corresponding standard deviation, for each of the training scenarios. The calculation assumes that there are 20 pictures of the target in each testing file. Approximately 50% of the target pictures in the testing set, was seen by the networks during training. The entries of table 4 are based on the table 12, table 13 and table 14 in appendix B. Figure 16 contains a diagram of the trinery and binary errors as function of training set size.

TABLE 4. False rejects

Errors		VsGr	Vs Some	VsGr Some5	VsGr Some10	VsGr Some15	Vs5%	Vs 10%	Vs 20%
Binary	**Mean**	7.2%	8.7%	10.3%	9.10%	9.6%	5.5	9.6%	10.1%
	Std	9.9%	10.1%	10.2%	7.0%	6.5%	4.7%	7.3%	9.1%
Trinery	**Mean**	6.1%	6.9%	4.2%	6.5%	7.3%	4.6%	8.7%	8.5%
	Std	8.2%	8.8%	5.0%	5.3%	5.7%	4.2%	5.5%	8.3%
Don't k	**Mean**	3.8%	5.7%	4.2%	3.2%	2.8%	2.3%	3.2%	5.2%
	Std	5.3%	8.9%	5.0%	4.05%	3.5%	4.7%	4.1%	5.1%

From table 4 we see that we have a large standard deviation. In Figure 15, we have plotted an histogram of the error frequency of two of the training scenarios. From this figure we see why we have a large standard deviation, and that our error distribution does not look like a Gaussian distribution

To investigate if there are any statistically significant difference in the ability of the networks to recognize unseen target pictures we performed the Kruskal-Wallis *(i. e. Bhattacharyya and Johnson, 1977, Chapter 15)* test for comparing *k* treatments.

This is a non-parametric test that does not require modeling a population in terms of a specific parametric form of density curve. We choose this test because we suspected a serious violation of the assumption of a specific distribution. The Kruskal-Wallis test is a test that gives each sample a rank, according to it's ordered size. In our case the samples are the numbers of false reject trinery errors for each trained network. (The values plotted in figure 15) Thus we have (37 X 4) +(11 X 4) = 192 samples of value 0 to 9. The exact sample values can be found in table 12 and table 13 in appendix B. Ordered size means that the samples were sorted according to their numeric value. All the different values were then given a rank. E. g. if the Zero valued samples occupied the first 57 places, their rank would be (57 +1) /2 =29, and if the next 59 places were occupied by ones, their rank would be

(57 + (59 +1)/2) =87. etc. The rank-sum for each treatment, in our case training scenarios included in the test, is then computed. The rank sum, is the sum of all the ranks assigned to each of the samples in each training scenario. If the rank sums for each treatment, divided by the numbers of samples in each treatment are about the same, there is no difference between the treatments. If the rank sums differ a lot, there is a difference between one or more of the treatments. The rank sums are converted to a χ^2 distribution, and the inference is done based on a χ^2 table. If a difference is detected, the population with the highest rank sum is centered to the right of the other populations. In our case, this indicates lower generalization.

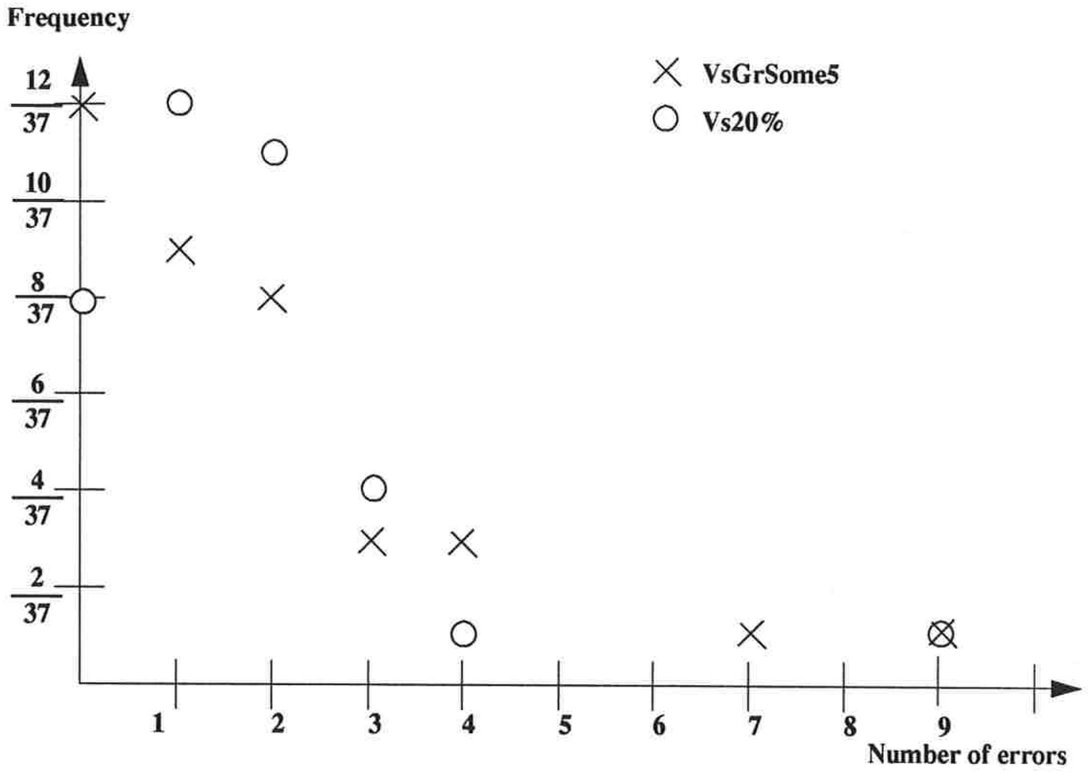


FIGURE 15. Histogram of false rejects, trinery errors. The frequency on the vertical axis refers to how many of the 37 networks had the number of errors specified on the horizontal axis, e. g. 12 networks had zero false reject errors in the VsGrSome5 training scenario.

The following hypothesis were tested on a $\alpha = 0.05$ confidence level:

H_0:All k continuous population distributions are identical

H_1: Not all k distributions are identical.

This test would indicate if the number of false rejects in each network, are about equally distributed. Table 5 summarize the results of the testing. The first column indicate what training scenarios were in the test set, and the second column indicate the outcome of the testing. The entries in the first column are ordered in terms of their rank sums, the training scenario to the left of the column is the entry with the lowest rank sum, and the right most training scenario had the highest rank sum.

TABLE 5. Inference about distributions, false rejects

H_1**:The distributions of the following training scenarios are not all identical**	H_0 **was rejected**
Vs5%, VsGr, VsSome, VsGrSome10, VsGrSome5, VsGrSome15, Vs20%, Vs10%	True
Vs20%, Vs10%	False
Vs5%, VsGr, VsSome, VsGrSome10, VsGrSome5, VsGrSome15	False

The Kruskal-Wallis test indicated that:

- Networks that have seen more different faces generalize worse.

The Vs5%, Vs10% and Vs20% training sets are constructed by randomly selecting pictures, that each have a probability of respectively 0.05, 0,1 and 0,2 to be chosen. If there are 20 pictures of each person then the average value of pictures selected of each person is 1, 2 and 4. That is each of these training scenarios are represented by, on the average, one, two or four pictures of each person. Whereas the VsGr, VsSome, VsGrSome5, VsGrSome10 and VsGrSome15 are represented by five pictures of respectively 24, 70, 95, 165 and 269 different people.

The test showed that the way the training scenarios were constructed did make a difference, and that the more different people the network has seen during training, the less it was willing to accept unseen target pictures during testing.

The tests were based on the results of 37 networks for each of the following training scenarios: VsGr, VsSome, VsGrSome5 and Vs20%, and on the results of 11 networks for the remaining four training scenarios. The error rate as a function of the size of the training set is plotted in figure 16.

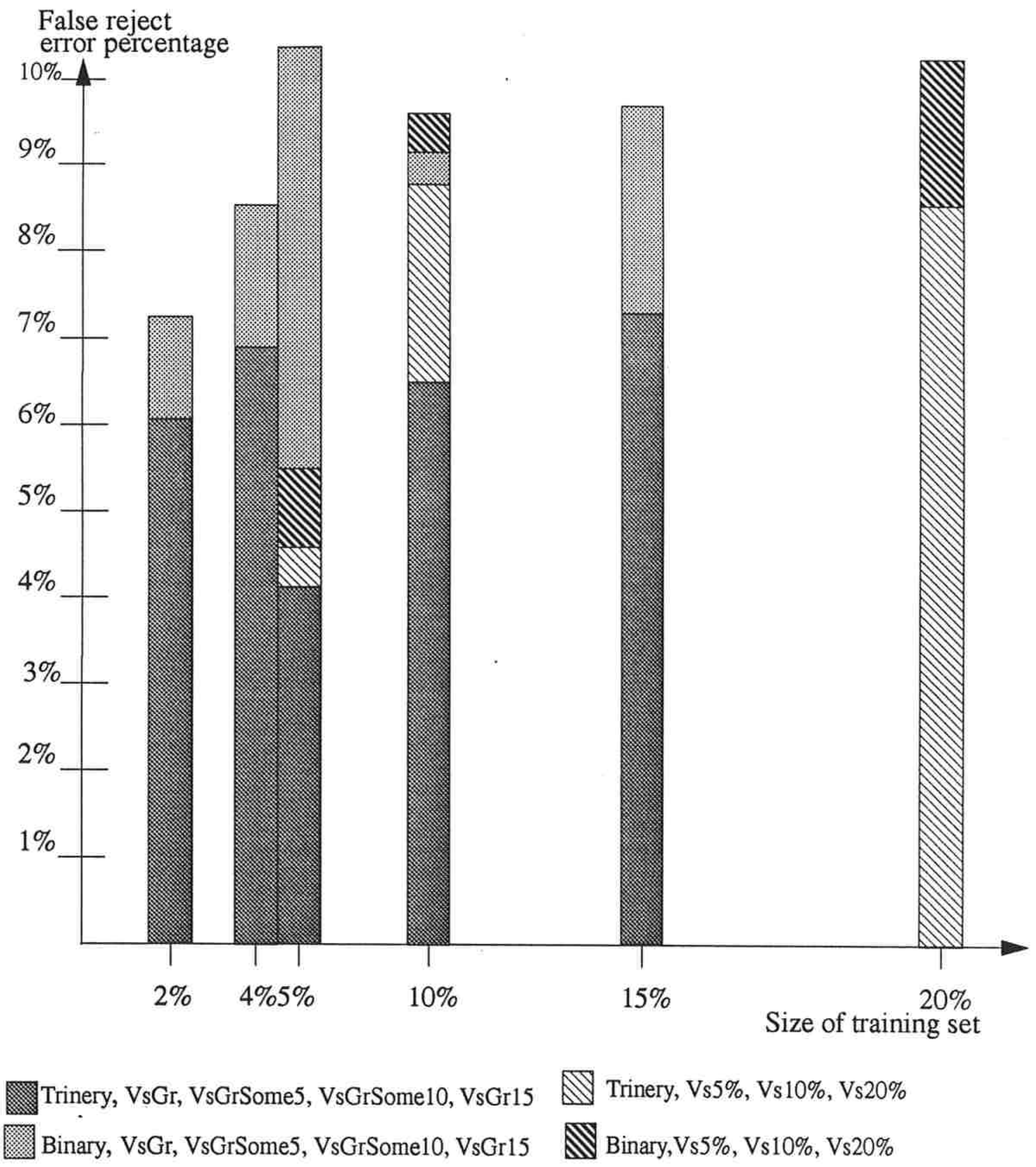


FIGURE 16. False rejections, trinery and binary errors as a function of size of training set.

8.4.2 The Pictures Causing False Rejects.

In this section we divided the pictures that were falsely rejected into groups. The groups where of kind profile pictures, cropped picture etc. This section also mentions a problem we had with target pictures that were zoomed differently. This section conclude that if the system is limited to only accept pictures of the target facing the camera, a false rejection rate of 1%-2% can be obtained, with the architecture we are using. We also conclude that small changes in facial expressions, glasses, and hair style do not cause any recognition problems.

To investigate what pictures caused the errors, we divided the false rejects, for the trinery error case, into four categories: Profile, up/down, crop, and other errors. The up/ down errors are errors caused by the target looking high up, so that the chin dominates the picture, or the subject is looking down so that the top of the head dominates the picture. The crop errors are errors caused by parts of the targets face being cropped off, or missing from the picture. The other errors are errors caused by other explainable things, such as dark sunglasses, a hand in the picture or other things of similar nature. The normal errors are pictures that the network should be able to classify correctly, in the opinion of the experimenters. The major source of this type of errors was caused by the target subject changing position, and the change was not reflected in the training set. Table 6 summarize the percentage of the different error types, for the different networks. The bottom row in the table says how many networks were trained using the training scenario specified at the top of each column. The percentage was calculated based on the total number of target pictures in each training file. The first entry in the table can be read as follows: 2.0% of the unseen target pictures were rejected because of profile picture, when the training scenario was that of VsGroup. The results in table 6 are plotted in figure 17.

TABLE 6. False rejects errors

	VsGr	**VsSome**	**VsGr Some5**	**VsGr Some10**	**VsGr Some15**	**Vs5%**	**Vs10%**	**Vs20%**
Profile	2.0%	2.8%	3.3%	3.2%	2.7%	1.8%	3.6%	3.6%
Up/Down	1.7%	2.0%	2.3%	2.3%	2.3%	1.8%	2.7%	2.2%
Crop	0.4%	0.4%	0.7%	0.5%	0.9%	0.9%	0.9%	0.8%
Others	0.9%	0.5%	0.8%	0.5%	0.9%	0%	0.9%	0.4%
Normal	0.9%	1.1%	1.3%	0%	0%	0%	0.5%	1.5%
	5.9%	6.8%	8.4%	6.5%	6.8%	4.5%	8.6%	8.5%
Total # of targets	37	37	37	11	11	11	11	37

From table 6 we see that the majority of errors are caused by profile pictures or pictures where the subject is looking up or down. If we in addition add the cropped pictures, we are left with an error rate of between 1% and 3%.

There may be many reasons why the networks had problems with the profile and up/down pictures. Below we have listed the two reasons we find the most likely.

- Not enough of these angles were represented in the training file.
- The decision region became dis-continuous, and networks with one hidden layer are unable to map dis-continuous decision regions.

Our training sets are composed randomly, and there is no guarantee that any profile or up/down pictures of the target was included in the training file. Between 20% and 30% of the pictures in the data base are of this kind.

We did not see any recognition problems due to glasses on or off, or by the subject changing hair style during the photo session, or any problems with the many facial expressions we photographed.

An example of the recognition performance of a network is as follows: a Vs10% network was trained with the following pictures in Figure 8b, 1, 2, 3, 7, 11, 13, 15, 17 and 20, and tested against 4, 5, 6, 8, 9, 10, 12, 14, 16 and 18. The testing pictures were all recognized, except for pictures 4 and 6.

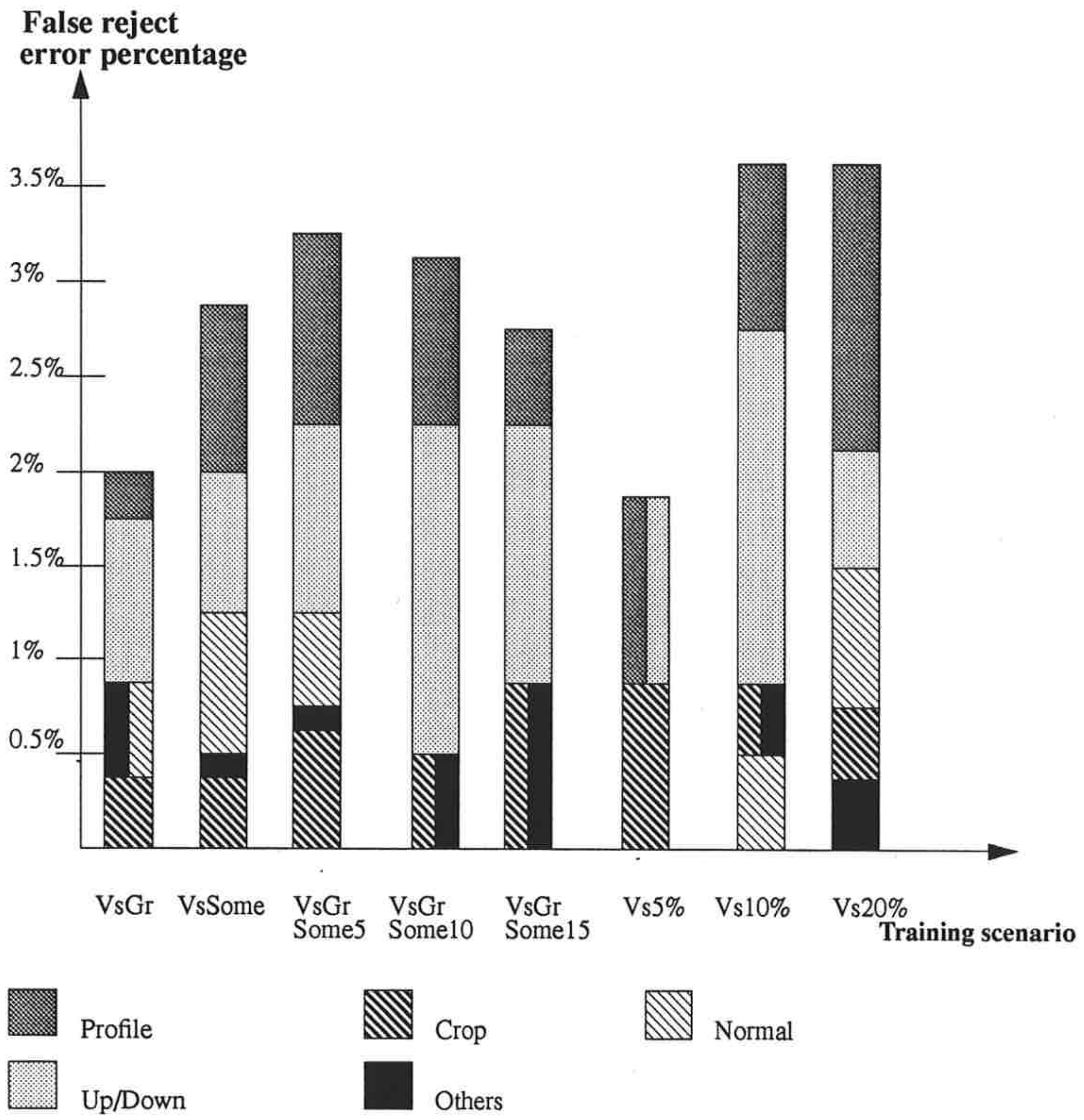


FIGURE 17. False reject trinery error types.

Not included in the numbers above is a test where we changed the scaling of three subject, that is the size of the subject, compared to the background. The network had no success in recognizing the targets. The network was in fact unable to learn the training set. Probably because there was a big difference in the size, and this caused a discontinuous decision region. This may be solved by adding another layer, or by using a range of different sizes, instead of just two different seizes. Figure 18 illustrates the difference in zooming.

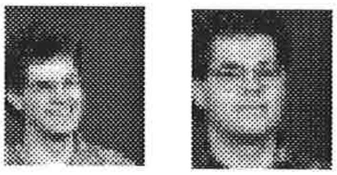


FIGURE 18. Difference in zooming

To conclude this section, we find that a false rejection rate of 1%-2% can easily be obtained, if the target is facing the camera.

We also found that the number of people seen during training does affect the networks ability to make generalizations about unseen target pictures.

8.4.3 False Accepts

In this section we investigate the false accept errors. We did a survey, using a non parametric inference, to test if the composition of the training sets had any effect on the performance. We found that the composition of the training sets did have an effect, and that the training scenario that had seen the most different pictures, yield the best performance. The section is concluded by stating that we got an indication that it is possible to train a network that rejects people that are not seen by the network during training.

Table 7 gives a summary of the percentages of the false accepts. The percentages are calculated based on the number of non target pictures in the testing set. The number of unseen pictures in the testing sets vary from 2% to 20% of the total number of pictures in the training set. If we calculated the percentage based on the number of unseen pictures, this only cause minor changes. We have chosen to do the analysis based on the total number of pictures because there is no guarantee that the network classify all the pictures in the training set correctly. A diagram of the binary and trinery errors in table 7 are plotted in figure 20 as a function of training set size.

TABLE 7. Average false acceptance under each training scenario.

	VsGr	VsSome	VsGr Some5	VsGr Some10	VsGr Some15	Vs5%	Vs10%	Vs20%
Binary	3.29%	0.62%	0.42%	0.30%	0.23%	0.23%	0.14%	0.04%
Std	3.60%	0.50%	0.40%	0.29%	0.24%	0.15%	0.12%	0.06%

TABLE 7. Average false acceptance under each training scenario.

	VsGr	VsSome	VsGr Some5	VsGr Some10	VsGr Some15	Vs5%	Vs10%	Vs20%
Triery	1.73%	0.18%	0.10%	0.13%	0.09%	0.08%	0.04%	0.02%
Std	2.27%	0.20%	0.14%	0.14%	0.14%	0.08%	0.04%	0.05%
Don't k.	2.87%	0.91%	0.66%	0.35%	0.29%	0.32%	0.20%	0.08%
Std	2.36%	0.65%	0.56%	0.27%	0.27%	0.19%	0.17%	0.07%

Again we note the large standard deviation, the frequency histogram in figure 19 suggests that we do not have a normal distribution.

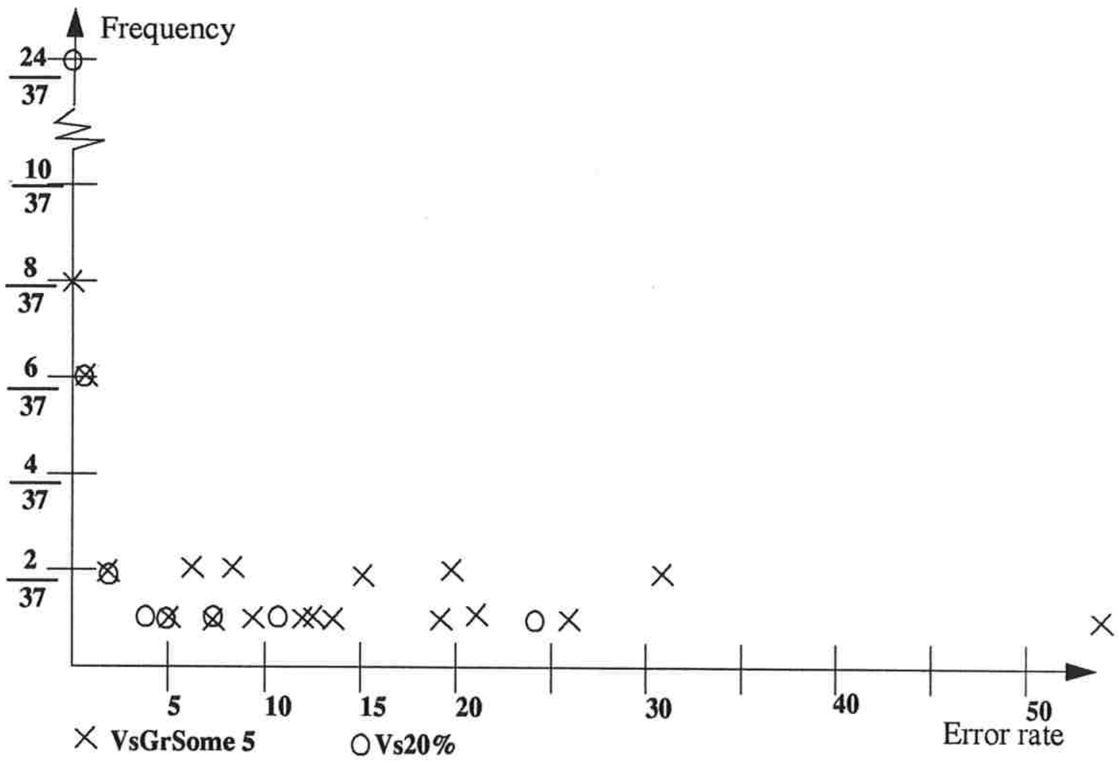


FIGURE 19. False accept trinery histogram. The frequency indicates how many of the networks had the number of errors indicated on the vertical axis.

We wanted to investigate if there are any statistically significant difference in the false acceptance performance for the different training scenarios. From figure 19, we may suspect either that the distribution approaches an exponential distribution, or that the error rate goes to zero, when the size of the training file increases. Because of the uncertainty about the underlying error distribution, we choose to do a nonparametric inference for comparing the performance of the different training scenarios.

We chose the Kruskal-Wallis *(i. e. Bhattacharyya and Johnson, 1977, Chapter 15)* test for comparing *k* treatments. (The detailed approach is explained in section 8.4.1)

The following hypothesis was tested:

H_0:All k continuous population distributions are identical

H_1: Not all k distributions are identical. .

By testing these hypothesis we wanted to detect if the number of errors were equally distributed throughout the testing scenarios. I. e. if the numbers of networks that gave a 0 error output, was about equal, or if some of the training scenarios had more networks with higher error rates. The null-hypothesis was tested and rejected or retained at a $\alpha = 0.05$ confidence level. The hypothesis tested, and the result of the testing is summarized in table 8.

TABLE 8. Inference about distributions

H_1:The distributions of the following training scenarios are not all identical	**H_0 was rejected**
Vs20%, Vs10%, VsGrSome15, VsGrsome5, VsGrSome10, Vs5%, VsGrSome10, VsSome, VsGr	True
Vs20%, Vs10%, VsGrSome15, VsGrsome5, VsGrSome10, Vs5%, VsGrSome10,	True
Vs10%, VsGrSome15, VsGrsome5, VsGrSome10, Vs5%, VsGrSome10,	False
Vs10%, VsGrSome15, VsGrsome5, VsGrSome10, Vs5%, VsGrSome10, VsSome	True
VsSome, VsGr	True

Because the testing was done based on the results of 37 networks in the VsGr, VsGrSome5 and Vs20%, and based on 11 networks for the VsGrSome10, VsGrSome15, Vs5% and Vs10% training scenarios, the results of the test should only be used as indicators of tendencies.

The results in table 8 indicate the following:

- The performance of the networks trained with 5% or more of the data did significantly better than the networks trained with less than 5% of the data.
- The networks trained with 20% of the data did significantly better than the networks trained with less than 20% of the data.
- The number of different people seen during training did not seem to make any difference in false acceptance. The overall number of pictures in the training set did seem to play a significant role.

We note that the performance between the Vs10%, VsGrSome10, Vs5%, VsGrSome5 and VsgrSome15 did not differ significantly. This suggests that the number of pictures in the training file has a greater effect on the performance, than the number of different people represented in the training set. This also suggests that it is possible to construct a system that rejects people not seen by the network during training.

We conclude this inference by stating that to get a low false acceptance rate when verifying human faces, the network needs to have seen a large variety of different pictures. The trinery and binary errors are plotted as a function of training set size in Figure 20.

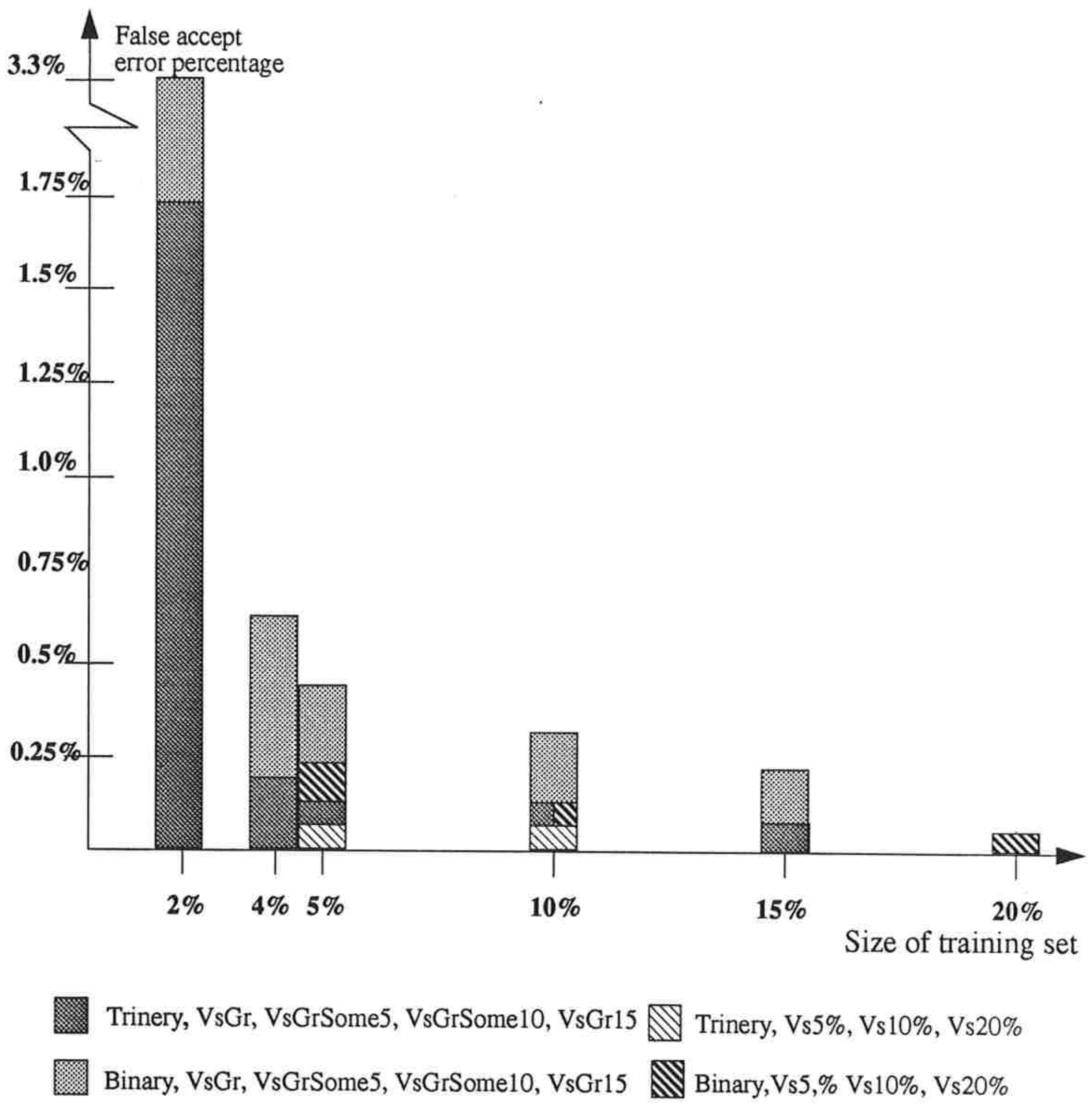


FIGURE 20. False accepts as function of training set size, trinery and binary errors.

8.4.4 The Pictures Causing False Acceptance.

In this section we investigated the particular pictures that were falsely accepted. We divided them into categories of pictures facing the camera, and pictures taken from other angels, or differed from the normal pictures in any other way. This section conclude that if only pictures facing the camera are allowed, an false acceptance rate of 0.01% can be obtained. We also found that the demographics of the people causing the errors match the demographics of the data base very well.

To investigate if any particular type of pictures caused the errors we divided the false accepts errors into seven different categories: Profile, up/down, side crop, bottom crop, black, other and normal. The side-crop are pictures where parts of the face is cropped of on either side. The bottom crop are pictures where the bottom part of the picture is missing, an area filled with zero's due to an error done while the pictures were taken. Black, is an empty picture, other is other explainable errors like dark glasses, hats, or hands present in the picture. Normal pictures are pictures that were mistaken for no obvious reason. Further we have divided the errors into two groups. The normal pictures that were falsely accepted and the other pictures that were falsely accepted. For each of these groups a survey was done on how many of these came from a similar demographic group, that is how many of the falsely accepted pictures were of people with similar skin tone and hair color. In table 9 this is denoted as % Sim. Dem. Many of the pictures that were mistaken in one of the eight networks trained for each target person, where mistaken in one or more of the other seven networks. This is denoted as %Seen. The survey was done for the trinery errors for the eleven targets, that we trained all eight networks for. We did not include the false accept errors in the VsGroup networks because these networks did much worse than any of the other networks trained in the other scenarios. The table contains the actual number of mistakes. The last row in table 9 indicates the total number of testing pictures that these numbers are based on. A diagram of the normal errors and the total of other errors is plotted in figure 21.

TABLE 9. False accept errors

	VsSome	**VsGr Some5%**	**VsGr Some10%**	**VsGr Some15%**	**Vs5%**	**Vs10%**	**Vs20%**
Profile	37	18	26	17	29	15	7
Up/Down	14	5	4	1	6	1	0
Side crop	18	11	9	1	1	2	0
Bottom Crop	9	8	9	7	11	3	2
Black	3	1	1	1	0	1	0
Other	3	2	1	1	3	5	2
Total, other	84	45	50	28	50	27	11
% Seen	58%	92%	84%	100%	59%	69%	75%
% Sim Dem	31%	26%	34%	50%	39%	27%	34%
Normal	101	27	91	69	38	22	20
% Seen	85%	100%	97%	80%	48%	89%	100%
% Sim Dem	58%	67%	79%	92%	64%	77%	100%
Sum	185	72	141	97	88	49	31
Based on	103 851	103 851	112 211	112 211	112 211	112 211	103 851

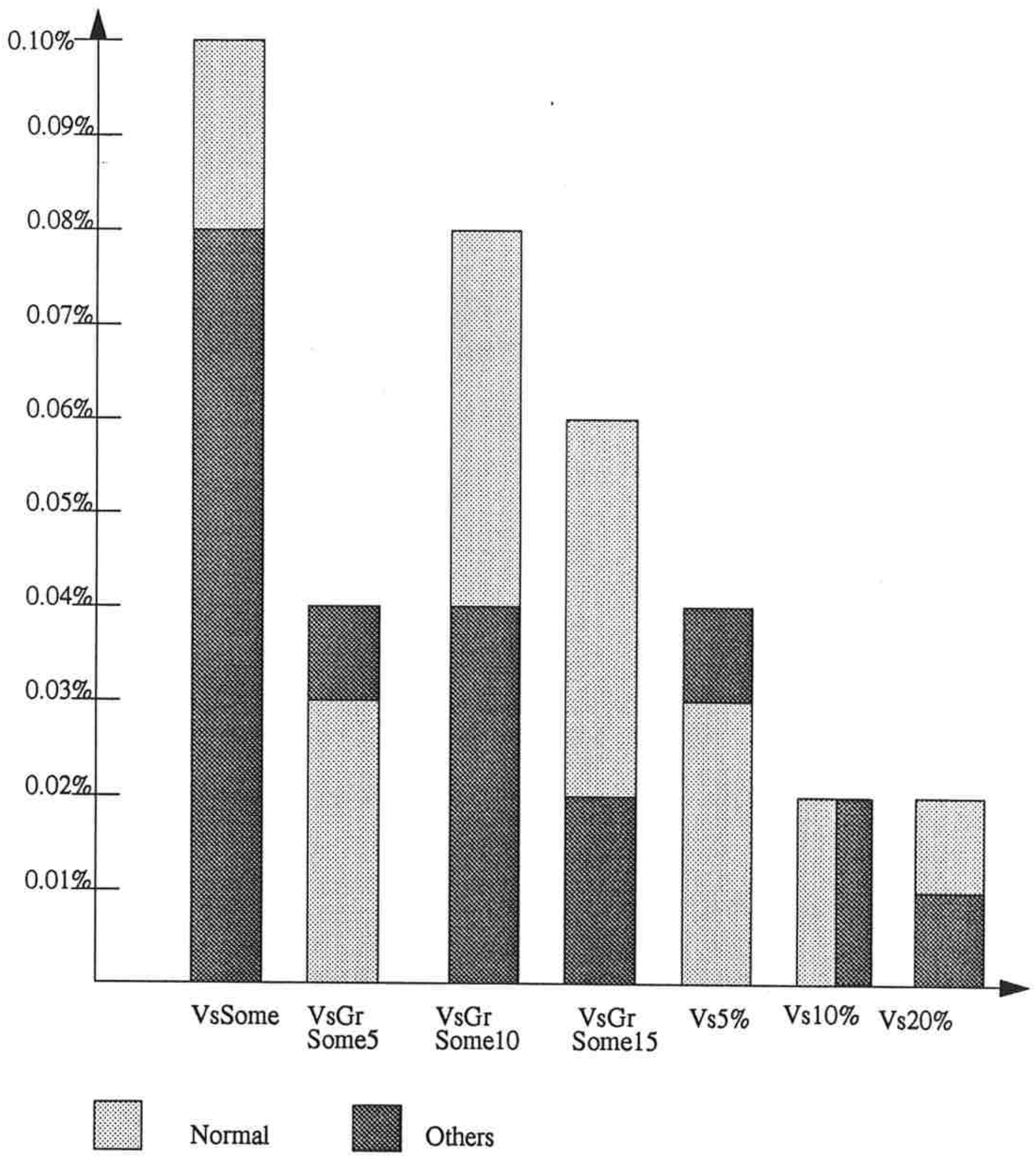


FIGURE 21. False accepts error types.

Table 9 indicates that if we have a system, that does not allow other pictures than pictures taken while facing the camera, we can obtain a false acceptance rate as low as 0.01%.

Table 9 indicates some tendencies that we would like to note.

- For all of the training scenarios the percentage of similar demographics were higher for the normal pictures than for the other pictures.

77 percent of the normal error pictures came from similar demographic groups to the target persons demographic group, whereas in the other error group, only 34 percent came from a similar demographic group. This is plotted in Figure 22.

- The percentage of similar demographics increased as the as the number of errors went down, and the number of subjects in the training file increased.

This indicates that errors caused by similar looking people will be the hardest errors to get rid of. This is also a good indication that the network is actually doing verification based on the face or other characteristics of the subject. Our results did not indicate, as we had hoped, that adding more pictures of similar looking people to the training set would tighten the decision boundaries, and confuse less of the normal pictures from similar demographic groups.

- There is also a tendency that the ratio between normal pictures errors and other errors increase, as the number of pictures in the training file increase.

This is an indication that the decision boundaries are getting tighter, and more complete, the more pictures the network sees. One may get rid of the confusion of strange pictures by letting the network see more of them during training.

The only plausible explanation we can have of when it comes to the acceptance of black pictures, pictures with hats, partially covered, or badly cropped faces, or other unfamiliar objects, is incomplete and open decision boundaries around the target person.

- The percentage of pictures seen by more than one network for each target person was slightly higher for the normal pictures (86%) than the pictures creating the other errors(77%). This is plotted in Figure 22.

The number of repeat errors was in general very high, 81%. This indicates that all seven training scenarios did a classification based on basically the same things. There was a difference in the decision boundaries, that keeps out some errors that were accepted in other networks.

Last we did a demographic test on of the 74 persons that had been falsely accepted in one or more of the networks to see if there was any particular group the network had problems with.The demographics in table 10 matches the demographics of the data base given in table 2.

TABLE 10. Demographics of subjects causing errors.

Sex	male:50%	female 50%	
Skin	light: 68%	medium: 32%	
Hair	Blond: 11%	brown: 78%	black 30%
	Grey: 15%	red: 3%	bald: 1%

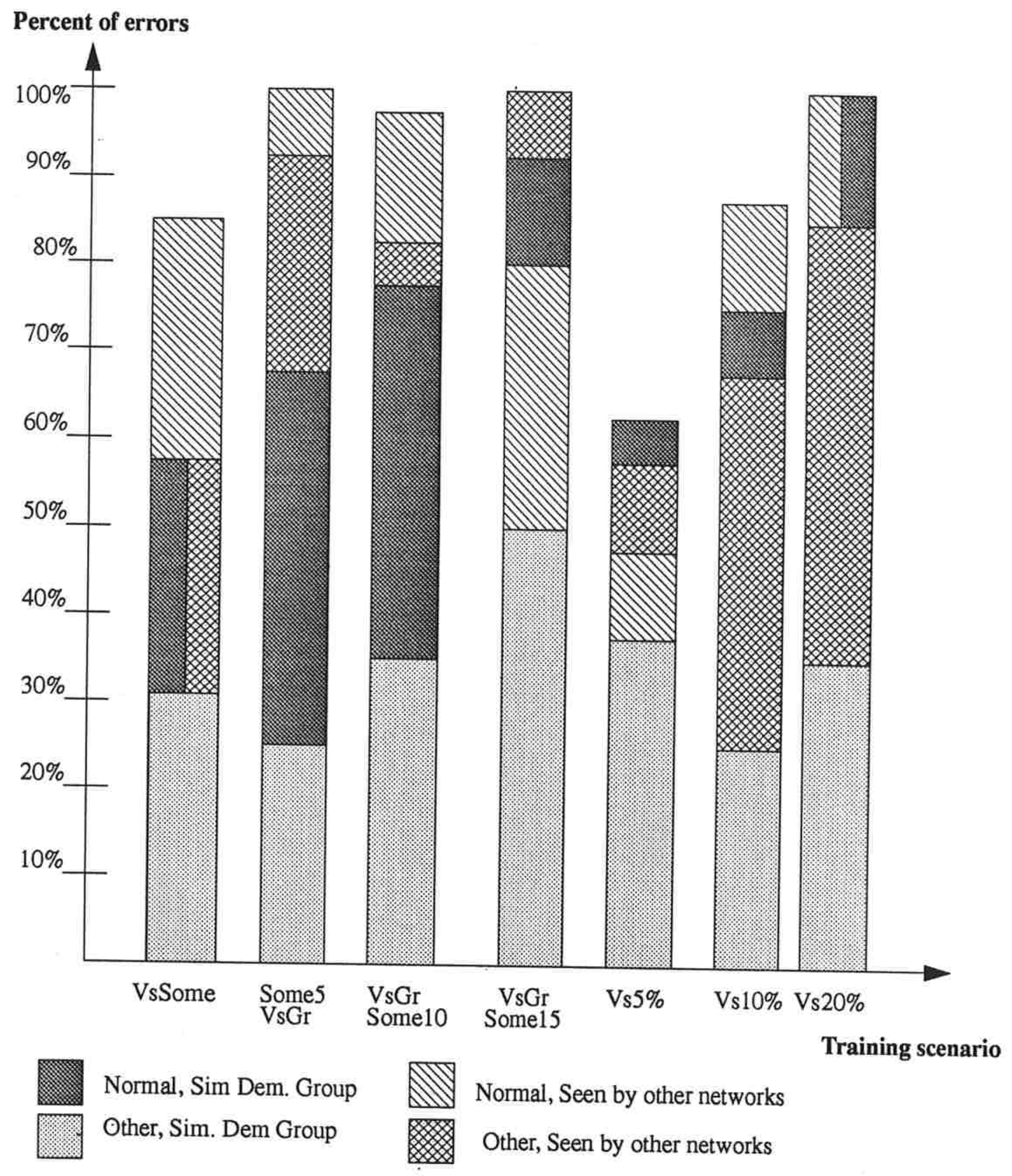


FIGURE 22. False accepts error types.

8.5 Comparison With Other Biometric Verification Systems

In this section I will briefly review the performance of five personnel verification systems, that it is natural to compare a facial verification system with. These devices, like the face recognition system are based on that user identify him/her self, and then presenting a feature, with the result of rejection of acceptance.

An evaluation of five verifiers was conducted (*Maxwell, 1987*), at the Sandia national laboratory, New Mexico. 80 people were enrolled in the project, and encouraged to attempt verification on each of the five devices several times a day during the test period, 60 of the enrolled subjects did this.

The five devices were the following:

A hand profile verifier by Recognition Systems, Inc.

A voice verifier by Voxtron Systems, Inc.

A voice verifier by AT&T Technologies

An eye retinal pattern verifier by Eyedntify, Inc.

A fingerprint verifier by Index, Inc.

The performances of the devices are summarized in table 11. Verify attempts refer to the number of trials of people claiming correct identity, the imposter attempts are attempts by people claiming false identity. Enrollment time, is the time spent to record the features used in the verifier, verification time is the time the user spent at the device, that is the time it took the user to identify him/her by entering a pin code, presenting the feature, and getting the result, accepted or rejected. The Imposter attempts experiment was done once, the users enrolled, identified them selves as the other users, and entered their own features. The verify attempts were recorded during a four month period. If the user made a mistake, e. g. entered the wrong pin code, or the feature was not correctly recorded, this was noted, and the data from these attempts are not included in the table below. In the last column, face picture, we have included the corresponding numbers from our survey, the Vs20% training scenario. The Verify attempts, are the ten unseen target pictures for each of the 37 network, and the imposter attempts are all the unseen, non target pictures. The verification time is the expected time of entering a code, and standing in front of a camera. The percentage of false rejects, is based on rejection of normal pictures.

TABLE 11. Performance of verifiers

	False	Hand Profile	Voice Pattern, AT&T	Vice Pattern Voxtron	Retinal Pattern	Finger print	Face picture
Verify attempts		1491	3206	2564	3082	3384	370
Imposter attempts		4055	3415	3795	5027	4849	267769
Enrollment time		54 sec	18 sec	144 sec	126 sec	114 sec	180 sec
Verification time		4.4 sec	8.8 sec	10.1 sec	7.5 sec	9.8 sec	10 sec
1'st try	**reject**	0.9%	12.5%	17.0%	10.8%	9.1%	1.5%
	accept	0.4%	0.1%	0.6%		0%	0.02%
2'nd try	**reject**	0.1%	3.3%	7.0%	4.8%	3.6%	
	accept		0.4%	0.9%			
3'rd try	**reject**	0.03%		5.2%	2.3%	1.8%	
	accept			0.9%			

This comparison, show that we have been able to obtain both lower false acceptance rate and false rejection rates. There is however no guarantee that the false rejection rate will not change, if pictures taken over a long period of time had been presented to the network. The false acceptance rate however, is well tested, and we do not expect this to change.

9.0 Conclusion

I will conclude this thesis with a summary of the face recognition project.

With our survey we found that:

- It is possible to get a correct verification of the target subject in 98% of the cases.

This is possible if only target pictures facing the camera are used. We got some recognition of profile, up, and down pictures, but more work needs to be done to get a high recognition rate of these angles. Possible approaches to solve this task is to add one more layer to the network, or by adding many more pictures of these angels to the training set.

We found that the larger number of different people were represented in the training file, the less willing the networks were to recognize the target pictures.

Before this facial verification system can be implemented into a usable system, a survey of returning subjects needs to be carried out. This to ensure that a high recognition rate can be retained over time.

- A false acceptance rate of order 0.01%, can easily be obtained.

This low number shows that a personnel verification system based on facial recognition can compete with other existing personnel verification systems.

We found that the number of different pictures the network had seen had a greater effect on the performance than the number of different people seen.

Both the above mentioned findings indicate that the networks needs to see a large variety of different people and pictures to be able to create tight decision boundaries around the target class. Tight decision boundaries are necessary to avoid errors that verify non target persons as the target.

These conclusions are based on 192 networks trained to verify the identity of 37 subjects. The networks were trained using eight differently composed training sets, and tested on all the images in our data base. By November 1991, our data base consisted of 10 221 pictures of 511 different subjects.

The results of the present project are much better than the results of facial recognition projects based on conventional pattern recognition techniques. Our results supports the results of the other facial recognition projects based on neural networks, these indicated that neural networks can be trained to recognize human faces.

This project shows that the backpropagation algorithm can perform pattern recognition tasks, using minimally pre processed images as patterns. The network will organize itself, and find characteristics within the data to base the classification on.

10.0 References

Anderson, J. A., "A memory storage model utilizing spatial correlation functions", Cybernetic, Vol5, pp 113-119, 1968.

Angeniol, B., G.DeLaCroix Vaubois, and J.Y. LeTexier, "Self-organizing feature maps and the traveling salesman problem", Neural Networks, vol. 1, pp 289-293, 1988.

Bhattacharyya, G. K., R. A. Johnson, "Statistical concepts and methods", John Wiley & Sons, Singapore 1977.

Blum, A. and R. L. Rivest, "Training a 3-node neural network is NP-complete," Proc. Computational Learning Theory (COLT), pp9-18, Morgan Kauffmann, 1988.

*Bradley, J. N. and C. M. Brislaw*n, "Image Compression by vector quantization of wavelet coefficients", Los Alamos National Laboratory, LA-UR-91-2225, 1991.

Bryson, A. E., and Y-C Ho, "Applied Optimal Control", [Revised Printing of the 1969 edition], Hemisphere Publishing, New York, 1975.

Castain, R.H., "Effectiveness of neural networks for advanced security - the issue of unique identification", Draft Proposal, SST-11 Los Alamos National Laboratory, 1990.

Duba, R. O., P. E. Hart, "Pattern Classification and Scene Analysis ", John Wiley & Sons, New York, 1973.

Encyclopedia Britanica, Vol 3, 1968 edition.

Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks", Neural Networks, vol.2, no 3, pp 183 - 192, 1989.

Goldstein, A. J., L. D. Harmon, and A. B. Lesk, "Identification of human faces", Proceedings of IEEE, 59, No. 5, pp. 748-760, 1971

Goldstein, A. J., L. D. Harmon, A. B. and Lesk, "Man-Machine Interaction in Human-Face Identification", Bell Systems Technical Journal 51, no 2, pp 399-427, 1972

Harel, D., "Algorithmic: The spirit of computing", Addison Wesely, Wokingham, 1987

Harmon, L.D. , M. K. Khan, R. Lasch, and P. F. Ramig, "Machine Identification of Human Faces ", Pattern Recognition, 13, no 2, pp 97-110, 1981

Hecht-Nielsen, R., "Neurocomputing", pp 14-19, pp 124-125, Addison - Wesley,Reading, MA, 1989.

Hebb, D., "The Organization of Behavior", Wiley, New York, 1949

Hertz, J. , A. Krogh, R.G. Palmer, "Introduction to the theory of neural computation", pp 6-8, Addison - Wesley, Redwood City, CA, 1991.

Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities", Proc. Nat. Acad. Sci. USA, vol.79, pp. 2554-2558, 1982.

Hopfield, J.J., "Neurons with a gradient response have collective computational properties like those of two-state neurons", Proc. Nat. Acad. Sci. USA, vol.81, pp.3088-3092, 1984.

Hush, D. R., and B. Horn, "An Introduction to Neural Networks", UNM Technical Report Number EECE 90-005, Department of Electric Engineering and Computer Engineering, University of New Mexico, 1990.

IJCNN, proceedings of, Seattle, WA, 1991.

Jacobs, R. A., "Increasing rates of convergence through learning rate adaption,", Neural Networks, vol. 1, pp295-308, 1988.

Jain, A. K. "Fundamentals of digital image processing", Prentice-Hall, chapter 10. 1989

Kaufman JR, J.G, and K. J. Breeding, "The Automatic Recognition of Human Faces from Profile Silhouettes", IEEE Trans. Sys. Man, Cyber, Vol 6,, pp 113-121, 1976

Kohonen,T. "An adaptive Associative Memory Principle", IEEE Trans. Comp., C-3, pp 444-445, 1974.

Kohonen,T., E. Oja, and P. Lehtio, "Storage and processing of Information in Distributed Associative Memory systems", In G. E. Hinton, J. A. Anderson (Eds), "Parallel models of associative memory", pp 123-125, 1981

Kohonen, T., "Self-Organization and Associative Memory", Springer Verlag,, pp 170, 1984.

Perry, J. L. , and J. M. Carney, "Human Face Recognition Using a Multilayer Perceptron Neural Network", Technical Report: Ensco-sas-tr-90-01, 1989.

Lein, M. Teknisk Ukeblad, "Nervepirrende vekst", No. 34 Vol. 138, yellow page 6, 1991.

Lippmann, R.P., "An Introduction to Computing With Neural Nets", IEEE ASSP magazine, Vol4. pp 4 -24, 1987.

Loyd, S. P., "Least square quantization in PCM", IEEE Trans. Info. Theory, vol. IT-28, PP. 129-137, 1982.

Mallat, S. G., "A theory for multiresolution signal decomposition: The wavelet Representation", IEEE Trans. Patt. Anal. Mach. Intel., Vol. PAMI-11, pp. 676-693, 1989.

Maxwell, R. L., "An identity verifier evaluation of performance", Sandia National Laboratories, New Mexico, USA, SAND-87-2279C, 1987.

McCulloch, W. S., and W. Pitts, "A logical calculus of the ideas immanent in the nervous activity", Bull. Math. Bio. Vol 5, pp 115-133, 1943.

Minsky, J. M., "Neural nets and the brain -Model problem", Doctoral Dissertation, Princeton University, Princeton NJ, 1954

Minsky, M., S. Pappert, "Perceptrons", MIT Press, Cambridge, MA, 1969.

O'Toole, A. J., R. B. Millward, and J. A. Anderson, "A Physical System Approach to Recognition Memory for Spatially Transformed Faces", Neural Networks, Vol 1. PP 179-199, 1988.

Pao, Yoh-Han, "Adaptive pattern recognition and neural networks", Addision - Wesley , Reading, MA, 1989.

Payne, T., I. Solheim and R. Castain, "Investigating Facial Verification Systems Using Backpropagation Neural Networks", Submitted for publication to Proc. IJCNN, Boston MA, 1992

Perry, J. L., and J. M. Carney, "Human Face Recognition Using a Multilayer Perceptron Neural Network", Proc. INNS Neural Net Conf., Wash, DC, Jan 1990, vol. II P413-416, 1990

Rabbani, M. ,P. W. Jones, "Digital Image Compression Techniques", SPIE Optical Engineering Press, 1991

Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," Psychol. Rev. Vol 65, pp 386-408, 1958.

R*umelhart, D. E., J. L. McClelland, and the PDP research group*, "Parallel Distributed Processing" Vo.1 and Vol. 2, MIT press, Cambridge, MA, 1986.

Solheim, I., T. Payne and R. Castain, "The potential in using Backpropagation Neural Networks for Facial verification systems", WINN - AIND, Aburn, AL, USA, 1992.

Takahashi, K., T. Sakaguchi, T. Minami, and O. Nakamura, "Description and matching of density variation for personal identification through facial images", SPIE Vol. 1360 Visual communication and image processing , pp 1694-1704, 1990.

Taylor, W. K., "Electrical Simulations of Some Nervous System Functional Activities", 1956, in C. Cherry [Ed], "Information Theory", pp 314-328, London: Butteworths, 1985.

Tou, J. T., R. C. Gonzales, "Pattern Recognition Principles ", Addision - Wesley, Reading, Massachusetts, 1974.

Von Neumann, J., "The general and logical theory of automata", in L. A.Jeffress,[Ed], "Cerebral Mechanism in Behavior", pp1-41; Wiley, New York, 1951.

Von Neumann, J., "Probabilistic logics and the synthesis of reliable organisms for unreliable components", in C. E. Shannon and J. McCarthy, (eds.), "Automata Studies", pp 43-98, Princeton University Press, Princeton NJ, 1956.

Widrow, B. and M. E. Hoff, "Adaptive switching circuits", 1960IRE WESCON Convention Record, pp 96-104, New York, 1960.

Widrow, B., "Generalization and information storage in networks of ADALINE neurons", in G. T.Yovitts [ed], "Self-Organizing Systems", Spartan Books, Washington, DC, 1962.

Widrow, B., and S. D. Stearns, "Adaptive Signal Processing", Prentice Hall, Englewood Cliffs, NJ, 1985.

Werbos, P. J., "Beyond Regression: New tools for prediction and analysis in the behavioral sciences", Doctoral Dissertation, Appl. Math., Harvard University, 1974.

White, H., "Learning in Artificial Neural Networks: A Statistical Perspective", Neural Computation, vol. 1, pp 425-464, 1989.

Appendix A

Compression program written in the language "C"

```
#include <stdio.h>
#include <math.h>
#include <sunvision/ip.h>

#define COMPRESS            5

unsigned char back1[50][50], back2[50][50];

main(argc, argv)
int argc;
char *argv[];
{
        int i, j, k, m, n, xleft, ytop;
        ip_Image *src, *crop, *scale, *comp, *comp2x;
        char cropfile[200], compfile[200], comp2xfile[200];
        char nwfiletrue[200], nwfilefalse[200];
        char subject[30], filename[30];
        double avg, back1avg, back2avg, slope, total_intens, xmom, ymom;
        float temp;
        unsigned char tmp, maxval;
```

```
FILE *fptrue, *fpfalse;

ip_init();
crop = ip_create(350, 400, 1, ipBYTE);
scale = ip_create(350, 400, 1, ipBYTE);
comp = ip_create((int)(350/COMPRESS), (int)(400/COMPRESS), 1, ipBYTE);
comp2x = ip_create((int)(175/COMPRESS), (int)(200/COMPRESS), 1, ipBYTE);

for (i=1; i<argc; i++)
{
        fprintf(stderr,"Working file %s\n", argv[i]);
        src = ip_load_file(argv[i]);
        if (src == NULL)
        {
                fprintf(stderr,"Could not load file %s\n", argv[i]);
        }
        else
        {
                ip_read(src, 35, 30, 50, 50, back1);
                ip_read(src, 480, 20, 50, 50, back2);
                back1avg = 0;
                back2avg = 0;
                for (j=0; j<50; j++)
                {

                        for (k=0; k<50; k++)
                        {
                                back1avg = back1avg + (float) back1[j][k];
                                back2avg = back2avg + (float) back2[j][k];
                        }
                }
                back1avg = back1avg / 2500.0;
                back2avg = back2avg / 2500.0;
                temp = back1avg + back2avg;
                total_intens = 0.0;
                xmom = 0.0;
                ymom = 0.0;
                for (j=0; j<450; j++)
                {
                        for (k=0; k<640; k++)
                        {
                                ip_getpixel(src, k, j, 0, (caddr_t)&tmp);
                                if ((float)tmp > temp)
                                {
                                        xmom = xmom + ((double)k * (double)tmp);
                                        ymom = ymom + ((double)j * (double)tmp);
                                        total_intens = total_intens + (double)tmp;
                                }
                        }
                }
                xleft = (int) (xmom / total_intens) - 125;
                if (xleft < 0) xleft = 0;
                if (xleft > 290) xleft = 290;
                ytop = (int) (ymom / total_intens) - 180;
```

```
                        if (ytop < 0) ytop = 0;
                        if (ytop > 80) ytop = 80;
                        maxval = 0;
                        for (j=xleft; j<(xleft+350); j++)
                        {
                                for (k=ytop; k<(ytop+400); k++)
                                {
                                        ip_getpixel(src, j, k, 0, (caddr_t)&tmp);
                                        if (maxval < tmp) maxval = tmp;
                                        ip_putpixel(crop, (int)(j-xleft), (int)(k-ytop), 0, (cad-
dr_t)&tmp);
                                }
                        }
                        avg = -1.0 * (back1avg + back2avg) / 4.0;
                        slope = 256.0 / ((float)maxval + avg);
                        ip_rescale(crop, scale, (float)avg, (float)slope, NULL);

                        strcpy(cropfile, argv[i]);
                        for (j=0; j<200 && cropfile[j] != '/'; j++) subject[j] = cropfile[j];

                        subject[j] = '\0';
                        j++;
                        for (k=0; k<30 && j<200 && cropfile[j] != '\0'; k++, j++) filename[k] = crop-
file[j];

                        filename[k] = '\0';

                        sprintf(comp2xfile, "mkdir /home/faces/%s 2> /dev/null", subject);
                        system(comp2xfile);
/*
                        sprintf(comp2xfile, "mkdir /home/faces/%s/crop 2> /dev/null", subject);
                        system(comp2xfile);
                        sprintf(comp2xfile, "mkdir /home/faces/%s/comp5 2> /dev/null", subject);
                        system(comp2xfile);
*/
                        sprintf(comp2xfile, "mkdir /home/faces/%s/comp10 2> /dev/null", subject); sys-
tem(comp2xfile);
/*
                        sprintf(comp2xfile, "mkdir /home/faces/%s/net5 2> /dev/null", subject);
                        system(comp2xfile);
*/
                        sprintf(comp2xfile, "mkdir /home/faces/%s/net10 2> /dev/null", subject);
                        system(comp2xfile);
/*

                        sprintf(cropfile, "/home/faces/%s/crop/%s", subject, filename);
                        ip_save_vff(scale, cropfile);

                        sprintf(compfile, "/home/faces/%s/comp5/%s", subject, filename);
                        sprintf(nwfiletrue, "/home/faces/%s/net5/%s.true", subject, filename);
                        sprintf(nwfilefalse, "/home/faces/%s/net5/%s.false", subject, filename);
                        fptrue = fopen(nwfiletrue, "w");
                        fprintf(fptrue,"%c %s\n", '!', nwfiletrue);
                        fpfalse = fopen(nwfilefalse, "w");
                        fprintf(fpfalse,"%c %s\n", '!',nwfilefalse);
                        for (j=0; j<400; j+=COMPRESS)
```

```
                {
                        for (k=0; k<350; k+=COMPRESS)
                        {
                                avg = 0;
                                for (m=0; m<COMPRESS; m++)
                                {
                                        for (n=0; n<COMPRESS; n++)
                                        {
                                                ip_getpixel(scale, (int)(k+n), (int)(j+m), 0,
(caddr_t)&tmp);
                                                avg = avg + tmp;
                                        }
                                }
                                avg = avg / (float)(COMPRESS*COMPRESS);
                                tmp = (unsigned char) avg;
                                ip_putpixel(comp, k/COMPRESS, j/COMPRESS, 0, (cad-
dr_t)&tmp);
                                avg = (avg / 128.0) - 1.0;
                                fprintf(fptrue, "%f ", avg);
                                fprintf(fpfalse, "%f ", avg);
                        }
                }
                ip_save_vff(comp, compfile);
                fprintf(fptrue, "1.0\n");
                fprintf(fpfalse, "0.0\n");
                fclose(fptrue);
                fclose(fpfalse);
*/
                sprintf(comp2xfile, "/home/faces/%s/comp10/%s", subject, filename);
                sprintf(nwfiletrue, "/home/faces/%s/net10/%s.true", subject, filename);
                sprintf(nwfilefalse, "/home/faces/%s/net10/%s.false", subject, filename);
                fptrue = fopen(nwfiletrue, "w");
                fprintf(fptrue,"%c %s\n", '!', nwfiletrue);
                fpfalse = fopen(nwfilefalse, "w");
                fprintf(fpfalse,"%c %s\n", '!',nwfilefalse);
                xleft = 2*COMPRESS;
                for (j=0; j<400; j+=xleft)
                {
                        for (k=0; k<350; k+=xleft)
                        {
                                avg = 0;
                                for (m=0; m<xleft; m++)
                                {
                                        for (n=0; n<xleft; n++)
                                        {
                                                ip_getpixel(scale, (int)(k+n), (int)(j+m), 0,
(caddr_t)&tmp);
                                                avg = avg + tmp;
                                        }
                                }
                                avg = avg / (float)(xleft*xleft);
                                tmp = (unsigned char) avg;
                                ip_putpixel(comp2x, k/xleft, j/xleft, 0, (caddr_t)&tmp);
                                avg = (avg / 128.0) - 1.0;
                                fprintf(fptrue, "%f ", avg);
```

```
                                        fprintf(fpfalse, "%f ", avg);
                                   }
                             }
                             ip_save_vff(comp2x, comp2xfile);
                             fprintf(fptrue, "1.0\n");
                             fprintf(fpfalse, "0.0\n");
                             fclose(fptrue);
                             fclose(fpfalse);
                             ip_destroy(src);
                       }
                 }
           }
```

Appendix B

Tables 12 and 13 contains the actual raw data from the experiment. Table 14 contains the number and percentage of pictures in each training file.

Gr1: Female, light blond hair, light skin.

Gr2:Female, medium brown hair, light skin.

Gr3:Female, light grey hair, light skin.

Gr4:Male, light brown hair, light skin.

Gr5:Female, medium brown hair, medium skin.

Gr6:Female, medium black hair, medium skin.

Gr7:Male, medium brown hair, light skin.

Gr8:Male, medium grey hair, light skin.

Gr9:Male, bald, light skin.

Gr10:Male, medium brown hair, medium skin.

Gr11:Male, medium black hair, medium brown skin.

TABLE 12. Tested on 9461 pictures of 477 people

Target#		VsGr		Vs Some		VsGr Some5		Vs20%	
Gr#		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
1	Binary	678	3	102	3	69	4	3	4
Gr1	Don't k	713	2	178	3	97	0	4	2
	Trinary	287	3	26	1	19	4	1	3
3	Binary	92	10	140	9	61	9	1	10
Gr2	Don't k	128	3	148	2	70	2	2	1
	Trinary	45	8	52	8	20	9	0	9

Target#		VsGr		Vs Some		VsGr Some5		Vs20%	
Gr#		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
29	Binary	936	0	146	0	93	0	1	2
Gr5	Don't k	586	0	168	0	134	0	0	2
	Trinary	562	0	37	0	15	0	0	2
35	Binary	26	3	33	3	26	3	7	1
Gr11	Don't k	46	0	48	0	46	0	13	0
	Trinary	2	3	2	3	2	3	4	1
52	Binary	65	2	94	1	67	3	5	4
Gr2	Don't k	113	1	94	2	109	0	17	2
	Trinary	21	2	32	1	6	3	0	3
65	Binary	767	2	87	3	84	3	0	3
Gr7	Don't k	602	0	123	0	123	3	2	1
	Trinary	436	2	44	2	20	2	0	3
74	Binary	309	1	54	1	24	1	7	2
Gr10	Don't k	334	0	81	0	49	1	10	2
	Trinary	126	1	18	1	5	1	1	1
100	Binary	400	0	30	0	23	2	2	1
Gr7	Don't k	442	0	73	1	50	1	6	0
	Trinary	192	0	3	0	1	1	0	1
104	Binary	125	1	44	3	58	3	2	4
Gr8	Don't k	116	1	70	3	41	3	8	3
	Trinary	64	1	4	1	31	1	0	3
145	Binary	7	0	40	0	3	1	0	2
Gr10	Don't k	55	1	55	1	8	1	0	1
	Trinary	0	0	5	0	0	0	0	1
150	Binary	174	0	90	0	47	0	3	0
Gr2	Don't k	198	0	85	0	77	2	3	0
	Trinary	52	0	49	0	9	0	5	1
152	Binary	1180	3	128	2	125	2	6	4
Gr4	Don't k	718	0	256	2	236	1	17	0

Target#		VsGr		Vs Some		VsGr Some5		Vs20%	
Gr#		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
	Trinary	728	2	24	2	31	2	0	4
155	Binary	192	0	131	0	77	2	2	2
Gr4	Don't k	267	0	177	0	103	0	13	1
	Trinary	75	0	41	1	21	2	0	2
160	Binary	60	0	16	0	14	0	0	2
Gr9	Don't k	74	0	46	0	50	0	3	1
	Trinary	21	0	2	0	1	0	0	1
186	Binary	597	6	56	7	31	7	4	3
Gr5	Don't k	645	2	129	0	83	0	11	1
	Trinary	249	5	16	7	1	7	1	3
211	Binary	41	3	43	4	20	4	3	4
Gr5	Don't k	35	1	46	1	14	0	7	1
	Trinary	23	3	26	3	15	4	0	3
220	Binary	189	1	11	2	6	2	5	1
Gr3	Don't k	235	1	33	4	26	2	8	1
	Trinary	60	1	0	1	0	1	0	0
222	Binary	548	2	1	2	1	2	4	0
Gr7	Don't k	544	2	11	0	12	0	5	3
	Trinary	236	0	1	2	1	2	0	0
225	Binary	54	0	41	0	36	0	27	0
Gr9	Don't k	44	0	58	0	49	0	14	1
	Trinary	34	0	9	0	8	0	24	0
228	Binary	256	1	50	2	44	2	15	2
Gr10	Don't k	142	2	56	2	57	1	17	1
	Trinary	182	0	21	1	13	2	11	2
244	Binary	20	1	5	1	5	1	2	1
Gr1	Don't k	40	0	12	0	10	0	2	0
	Trinary	7	1	0	1	0	1	0	1
284	Binary	246	0	33	1	10	1	1	1

Target#		VsGr		Vs Some		VsGr Some5		Vs20%	
Gr#		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
Gr8	Don't k	282	1	51	0	26	0	3	1
	Trinary	180	1	5	1	0	1	0	1
299	Binary	1469	2	5	2	4	2	0	2
Gr6	Don't k	764	0	23	0	14	0	0	1
	Trinary	978	2	0	2	0	2	0	2
304	Binary	313	1	12	2	3	1	1	2
Gr8	Don't k	281	0	45	6	13	0	0	0
	Trinary	141	1	0	1	0	1	1	2
305	Binary	24	1	12	3	11	3	0	1
Gr8	Don't k	30	2	19	2	17	2	5	0
	Trinary	8	1	0	1	0	1	0	1
307	Binary	93	3	8	5	5	6	1	1
Gr3	Don't k	105	4	16	8	9	2	7	0
	Trinary	36	2	1	3	1	4	1	1
318	Binary	66	1	49	1	19	1	0	1
Gr5	Don't k	51	1	51	0	27	1	3	0
	Trinary	35	1	21	1	8	1	0	1
323	Binary	292	2	77	2	17	4	5	2
Gr6	Don't k	260	3	120	1	44	0	12	1
	Trinary	157	1	17	2	0	3	0	2
340	Binary	102	1	14	1	14	1	3	1
Gr9	Don't k	151	0	31	1	28	1	14	0
	Trinary	26	1	1	1	1	1	0	1
351	Binary	393	2	89	2	68	2	4	3
Gr4	Don't k	427	0	183	1	141	1	5	0
	Trinary	150	2	20	2	12	2	2	3
406	Binary	415	0	70	0	59	0	1	1
Gr3	Don't k	393	0	105	1	105	1	5	1
	Trinary	222	0	15	0	6	0	0	1

Target#		VsGr		Vs Some		VsGr Some5		Vs20%	
Gr#		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
408	Binary	648	0	194	0	170	0	12	2
Gr4	Don't k	417	0	213	1	191	3	25	2
	Trinary	375	0	87	0	65	0	1	2
426	Binary	290	1	24	2	23	2	4	1
Gr1	Don't k	316	1	39	0	45	1	13	2
	Trinary	131	1	2	2	2	2	0	1
430	Binary	98	0	43	0	26	0	14	0
Gr11	Don't k	91	0	76	0	41	0	20	0
	Trinary	44	0	14	0	7	0	7	0
456	Binary	161	0	111	0	61	0	3	3
Gr2	Don't k	151	0	123	0	78	2	4	4
	Trinary	78	0	52	0	26	0	2	2
457	Binary	41	0	14	0	1	2	3	0
Gr11	Don't k	73	0	34	0	20	0	7	0
	Trinary	4	0	4	0	0	0	1	0
478	Binary	156	0	68	0	46	0	0	2
Gr6	Don't k	173	0	91	0	73	0	0	2
	Trinary	58	0	21	0	14	0	0	0
Sum	Binary	1 523	53	2 165	64	1 451	76	151	75
Mean		311.43	1.43	58.51	1.73	39.22	2.05	4.08	2.03
Std		340.10	1.97	47.35	2.02	37.53	2.04	5.34	1.82
Sum	Dont k.	10 042	28	3 167	42	2 316	31	285	38
		271.41	0.76	85.59	1.14	62.59	0.84	7.70	1.03
Std		223.04	1.06	161.26	1.78	52.50	0.99	6.35	1.01
Sum	Trinary	6 025	45	635	51	361	63	59	63
Mean		162.84	1.22	17.16	1.38	9.73	1.73	1.59	1.70
Std		213.38	1.64	19.26	1.75	13.16	1.97	4.37	1.65

TABLE 13. Results, tested on 10221

Target#		*VsGrSome10%*		*VsGrSome15%*		*Vs5%*		*Vs10%*	
		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
52	Binary	84	3	48	3	15	2	39	2
	Don't k	88	2	91	0	33	3	56	0
	Trinary	24	1	2	3	1	2	11	2
65	Binary	22	3	2	3	14	0	12	1
	Don't k	46	0	6	1	25	0	21	1
	Trinary	5	3	0	3	3	0	3	1
145	Binary	0	0	0	1	0	2	0	2
	Don't k	2	1	2	0	2	1	1	1
	Trinary	0	0	0	1	0	1	0	2
155	Binary	80	2	75	2	38	2	25	2
	Don't k	83	0	72	0	43	0	38	2
	Trinary	39	1	39	1	10	2	4	2
220	Binary	1	2	2	2	22	0	3	5
	Don't k	5	1	6	1	42	0	9	0
	Trinary	1	1	1	1	5	0	0	3
225	Binary	56	0	49	0	41	0	31	0
	Don't k	35	0	29	1	39	0	26	0
	Trinary	38	0	34	0	23	0	15	0
305	Binary	14	3	14	3	0	2	0	3
	Don't k	17	2	18	2	3	0	1	0
	Trinary	3	1	2	1	0	2	0	3
318	Binary	28	1	26	1	32	1	10	2
	Don't k	29	0	24	0	30	0	16	1
	Trinary	12	1	11	1	16	1	5	2
323	Binary	18	4	14	4	46	2	17	3
	Don't k	38	1	32	1	75	2	33	2
	Trinary	2	3	1	3	19	1	3	3
426	Binary	16	2	12	2	19	1	9	1

Target#		*VsGrSome10%*		*VsGrSome15%*		*Vs5%*		*Vs10%*	
		False accept	False reject	False accept	False reject	False accept	False reject	False accept	False reject
	Don't k	24	0	18	0	28	0	9	0
	Trinary	4	2	0	2	1	1	4	1
430	Binary	19	0	16	0	31	0	12	0
	Don't k	28	0	29	0	34	0	11	0
	Trinary	14	0	7	0	10	0	4	0
Binary	Sum	338	20	258	21	258	12	158	21
	Mean	30.78	1.82	23.45	1.91	23.45	1.09	14.36	1.91
	Std	29.35	1.40	23.99	1.30	15.62	0.94	12.67	1.45
Don't k	Sum	395	7	327	6	354	5	221	7
	Mean	35.91	0.64	29.73	0.55	32.18	0.45	20.09	0.64
	Std	27.84	0.81	27.84	0.69	19.83	0.93	16.99	0.81
Trinary	Sum	142	13	97	16	88	10	49	19
	Mean	12.92	1.30	8.82	1.45	8	0.91	4.45	1.73
	Std	14.50	1.06	14.15	1.13	8.23	0.83	4.68	1.10

TABLE 14. Numbers of subjects and pictures in each training file, and percentage of the total number of pictures in testing file

Target		Pictures of target	VsGr	Vs Some	VsGr Some 5%	*VsGr Some 10%*	*VsGr Some 15%*	*Vs5%*	*Vs10%*	Vs20%
1	Pictures	10	134	360	484					
	Percent	50%	1.42%	3.81%	5.12%					
3	Pictures	8	157	359	631					
	Percent	35%	1.66%	3.79%	6.67%					
29	Pictures	10	125	360	475					2063
	Percent	50%	1.32%	3.81%	5.02%					20.18%
35	Pictures	10	94	360	444					2034
	Percent	50%	0.99%	3.81%	4.69%					21.50%
52	Pictures	10	157	360	507	800	1483	543	1020	

Target		Pictures of target	VsGr	Vs Some	VsGr Some 5%	*VsGr Some 10%*	*VsGr Some 15%*	*Vs5%*	*Vs10%*	Vs20%
	Percent	50%	1.66%	3.81%	5.36%	8.46%	14.51%	5.74%	10.78%	
65	Pictures	10	147	360	497		1467	531	1055	
	Percent	48%	1.55%	3.81%	5.25%		14.35%	5.42%	11.15%	
74	Pictures	10	120	360	470					2014
	Percent	50%	1.27%	3.81%	4.97%					21.29%
100	Pictures	10	147	360	497					
	Percent	50%	1.55%	3.81%	5.25%					
104	Pictures	9	130	359%	480					
	Percent	45%	1.37%	3.79%	5.07%					
145	Pictures	10	120	360	470		1444	520	1009	2088
	Percent	48%	1.27%	3.81%	4.97%		14.13%	5.50%	10.66%	22.07%
150	Pictures	10	157	360	507					
	Percent	50%	1.66%	3.81%	5.36%					
152	Pictures	10	135	360	485					2049
	Percent	48%	1.43%	3.81%	5.13%					20.05%
155	Pictures	10	135	360	485	852	1455	501	1064	
	Percent	50%	1.43%	3.81%	5.13%	8.34%	14.24%	5.30%	11.25%	
160	Pictures	10	105	360	455					
	Percent	50%	1.11%	3.81%	4.81%					
186	Pictures	10	125	360	475					2093
	Percent	48%	1.32%	3.81%	5.02%					20.48%
211	Pictures	10	125	360	475					2024
	Percent	50%	1.32%	3.81%	5.02%					19.80%
220	Pictures	10	145	360	495	867	1470	505	1106	2038
	Percent	50%	1.53%	3.81%	5.23%	8.48%	14.38%			19.94%
222	Pictures	10	147	360	497					
	Percent	48%	1.55%	3.81%	5.25%					
225	Pictures	10	105	360	455	824	1427	550	1037	2033
	Percent	50%	1.11%	3.81%	4.81%	8.06%	13.94%	5.81%	10/96%	21.49%
228	Pictures	10	120	360	470					2046

Target		Pictures of target	VsGr	Vs Some	VsGr Some 5%	*VsGr Some 10%*	*VsGr Some 15%*	*Vs5%*	*Vs10%*	Vs20%
	Percent	50%	1.27%	3.81%	4.97%					21.63%
244	Pictures	10	134	360	484					
	Percent	50%	1.42%	3.81%	5.12%					
284	Pictures	10	130	360	480					
	Percent	50%	1.37%	3.81%	5.07%					
299	Pictures	10	117	360	467					2037
	Percenta	50%	1.24%	3.81%	4.94%					19.93%
304	Pictures	10	134	360	484					
	Percent	50%	1.42%	3.81%	5.12%					
305	Pictures	10	130	360	480		1432	548	1012	
	Percent	48%	1.37%	3.81%	5.07%		14.01%	5.79%	10.70%	
307	Pictures	10	145	360	495					
	Percent	50%	1.53%	3.81%	5.23%					
318	Pictures	10	125	360	475	847	1450	504	1042	
	Percent	50%	1.32	3.81%	5.02%	8.29%	14.22%	5.33%	11.01%	
323	Pictures	10	117	360	467	829	1422	522	1015	
	Percent	50%	1.24%	3.81%	4.94%	8.11%	13.91%	5.52%	10.73%	
340	Pictures	10	105	360	455					
	Percent	50%	1.11%	3.81%	4.81%					
351	Pictures	10	135	360	485					
	Percent	50%	1.43%	3.81%	5.13%					
406	Pictures	10	145	360	495					
	Percent	50%	1.53%	3.81%	5.23%					
408	Pictures	10	125	360	475					2032
	Percent	50%	1.32%	3.81%	5.02%					19.88%
426	Pictures	10	134	360	484	856	1459			
	Percent	50%	1.42%	3.81%	5.12%	8.37%	14.27%			
430	Pictures	10	94	360	444	816	1419	515		2065
	Percent	53%	0.99%	3.81%	4.69%	7.98%	13.88%	5.44%		21.83%
456	Pictures	10	157	360	507					

Target		Pictures of target	VsGr	Vs Some	VsGr Some 5%	*VsGr Some 10%*	*VsGr Some 15%*	*Vs5%*	*Vs10%*	Vs20%
	Percent	50%	1.66%	3.81%	5.36%					
457	Pictures	10	94	360	444					
	Percent	50%	0.99%	3.81%	4.69%					
478	Pictures	10	117	360	467					
	Percent	53%	1.24%	3.81%	4.94%					
Avg			129	360	482	836	1448	524	10430	2047