

NEURAL NETWORK BASED FACE DETECTION

ERHAN DOĞRUKARTAL

B.S., Computer Engineering, Işık University, 2007

Submitted to the Graduate School of Science of Engineering
In partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY

2010

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

NEURAL NETWORK BASED FACE DETECTION

Erhan Doğrukartal

APPROVED BY:

Assoc. Prof. M. Taner Eşkil (Işık University) _____

(Thesis Supervisor)

Assist. Prof. Tamer Dağ (Kadir Has University) _____

Assist. Prof. Boray Tek (Işık University) _____

APPROVAL DATE:

NEURAL NETWORK BASED FACE DETECTION

Abstract

One of the challenging problems in computer vision is face detection. The goal of face detection is to locate all regions that contain a face regardless of any three dimensional transformation and lighting condition. There are lots algorithms which were proposed for a solution of face detection problem. In this thesis, I have implemented a neural network based face detection approach. The system detects upright frontal faces in a given image. Some heuristics were used in order to reduce the number of false detections. Also more than one neural networks are used in order to improve the detection capability. There are lots of neural network based face detection systems in the literature. Since these proposed systems in the literature are complex they cannot be used for real time applications. First and the most important difference between the proposed system in this thesis and the systems in the literature is simplicity of a network. Another major difference is size of a window which is used for face detection. In this thesis 10×10 pixel size window which is the smallest window size in the literature is used for detection. Furthermore simple system and small size window gives a faster training and running.

YAPAY SİNİR AĞLARI İLE YÜZ BULMA

Özet

Yüz bulma problemi bilgisayarla görme konusunun en önemli problemlerinden biridir. Yüz bulmanın amacı verilen bir resimde yüz olup olmadığına yüzün 3 boyutlu transformasyonuna ve ışık şartlarına bakmaksızın karar vermektir. Yüz bulma problemi ile ilgili birçok önerilmiş çözüm mevcuttur. Bu çalışmanın konusu yapay sinir ağları ile yüz bulmadır. Bu çalışmadaki sistemin amacı verilen resimdeki yüzleri yapay sinir ağlarını kullanarak bulmaktır. Hatalı bulunan yüz sayısını azaltmak için bazı algoritmalar geliştirildi. Ayrıca yüz bulma yüzdesini yükseltmek için birden fazla yapay sinir ağı kullanıldı. Literatürde yapay sinir ağları ile yüz bulma konusunda birçok çalışma mevcuttur. Fakat bu çalışmaların genel özelliği verilen önerilen sistemlerin karmaşık olmasından dolayı gerçek zamanlı uygulamalarda kullanılamamalarıdır. Bu çalışmada belirtilen sistemin en önemli ve de literatürdeki diğer çalışmalardan ayıran başlıca özelliği basit bir sistem olması ve bu basitlik sayesinde hızlı olmasıdır. Diğer bir özelliği de yüz bulma işlemi sırasında kullandığı işlem pencere boyutu. Bu çalışmadaki sistemin kullandığı pencerenin boyutları 10×10 dur. Buda be çalışmadaki sistemi literatürdeki en basit sistem yapar. Ayrıca küçük pencere boyutu ve sistemin basit olması sistemin eğitilmesi için ve çalıştırılması için daha az zaman almasını sağlar.

Acknowledgements

There are many people who helped to make my years at the graduate school most valuable. First, I thank Assoc. Prof. Taner ESKIL, my major professor and dissertation supervisor. Having the opportunity to work with her over the years was intellectually rewarding and fulfilling. I also thank everyone who contributed much to the development of this research starting from the early stages of my dissertation work.

Many thanks to professors of the department of computer engineering, who patiently answered my questions and problems on word processing. I would also like to thank to my graduate student colleagues who helped me all through the years full of class work and exams. My special thanks go to Melih Sozdinler whose friendship I deeply value.

The last words of thanks go to my big family. I thank my parents Ali Osman and Nazen Dođrukartal , my brothers Erdal, Erkan, Ertekin and Hasan Dođrukartal and my sisters Sadegul and Elif Dođrukartal for their patience and encouragement.

I also thank to Tubitak for their support during this research.

To my parents Mr. Ali Osman and Mrs. Nazen Dođrukartal

Table of Contents

Abstract.....	III
Özet.....	IV
Acknowledgements	V
List of Figures.....	IX
List of Tables	XII
1. Introduction	1
2. Face detection.....	3
2.1 Challenges in Face Detection	3
2.2 Face Detection Methods.....	6
2.2.1 Knowledge – Based Top Down Methods	6
2.2.2 Bottom-Up Feature-Based Methods.....	8
2.2.3 Template Matching.....	12
2.2.4 Appearance-Based Methods	14
3. Neural Networks	21
3.1 Introduction to Neural Networks.....	21
3.2 Biological Neuron.....	22
3.3 Simulating the Biological Neuron.....	24
3.4 History of Neural Networks.....	25
3.5 Neural Network Types.....	27
3.5.1 Single-Layer Feedforward Neural Network.....	27
3.5.2 Multilayer Neural Network	28
3.5.3. Recurrent Neural Network	28
3.6 Solving Problem with Neural Networks	29
3.6.1 Problems can not be Solved by Neural Network	29
3.6.2 Problems can be Solved by Neural Network	30
3.7 Structure Of Neural Network.....	30
3.7.1 the Neuron.....	31
3.7.2 Neuron Connection Weights	32
3.7.3 Neuron Layers.....	32
3.8 Training the Neural Network.....	34
3.9 Backpropogation Algorithm.....	35
3.11 Validating the Neural Network.....	37

4. Neural Network Based Face Detection.....	38
4.1 Background.....	38
4.2 Description of the our System.....	42
4.2.1 Preprocess Step.....	42
4.2.2 Neural Network Stage.....	46
4.2.2 Postprocess.....	50
5. Neural Network Based Face Detection and Image Processing System	54
5.1 Introduction to Neural Network Based Face Detection and Image Processing System	54
5.1.1 Neural Network Part	55
5.1.2 Image Processing Part.....	58
6. Experimental Results	61
6.1 Face Threshold.....	62
6.2 Multiple Networks	68
Conclusion.....	73
References	74
Curriculum Vitae	77

List of Figures

Figure 2.1: Samples of different poses of human face.....	3
Figure 2.2: Samples of beards, mustaches and glasses.....	4
Figure 2.3 Samples of facial expressions.....	4
Figure 2.4 Samples of occlusion.....	4
Figure 2.5: Image orientation.....	5
Figure 2.6: Effects of illumination of face detection.....	5
Figure 2.7: Typical face used in knowledge-based top-down methods.....	6
Figure 2.8: Multi resolution hierarchy.....	7
Figure 2.9 Intensity image and edge map from Canny's edge detector.....	9
Figure 2.10 Skin Color test image and result image.....	11
Figure 2.11: Multiple Feature example.....	11
Figure 2.12 Eigenfaces.....	15
Figure 2.13: Architecture of Rowley.....	17
Figure 3.1: Neural network transforms inputs into outputs.....	21
Figure 3.2: A Neuron Cell.....	23
Figure 3.3: A Digital Signal.....	23
Figure 3.4: Single layer feed forward neural network.....	27
Figure 3.5: Multilayer feedforward neural network.....	28
Figure 3.6: Recurrent Neural Network.....	29
Figure 3.7: TanH activation Function.....	32
Figure 3.8: Neural Network Layers.....	34
Figure 3.9: Schematic diagram of supervised Learning.....	35
Figure 3.10: Schematic diagram of unsupervised learning.....	35
Figure 4.1: Algorithm used by Rowley for face detection.....	40
Figure 4.2: Filtering process using by Anifantis for face detection.....	41
Figure 4.3 Convolutional Neural Network structure.....	42
Figure 4.4: Original and histogram equalized images.....	44

Figure 4.5: Oval mask.....	45
Figure 4.6 Image pyramid.....	46
Figure 4.7: Image Scanning.....	47
Figure 4.8: Structure of the network.....	48
Figure 4.9: Multiple detections.....	49
Figure 4.10 Face samples used during training.....	50
Figure 4.11: Nonface samples used during training.....	51
Figure 4.12: Multiple detections with false detections.....	51
Figure 5.1: Neural Network Based Face Detection System.....	55
Figure 5.2: Using of create command.....	56
Figure 5.3: Using of train command.....	56
Figure 5.4: Using the run command with a file.....	57
Figure 5.5: Using run command with single image.....	57
Figure 5.6: Using run command with images under a specific folder.....	57
Figure 5.7: Using the save command.....	58
Figure 5.8: Using the load command.....	58
Figure 5.9: Using the histogram command.....	59
Figure 5.10: Result of the histogram command.....	59
Figure 5.11: Using the gamma command.....	60
Figure 5.12: Result of the gamma command.....	60
Figure 5.13: Usage of the gradient command.....	60
Figure 5.14: Result of the gradient command.....	61
Figure 6.1: Face Detection Rate against the threshold value.....	64
Figure 6.2: False Detection Rate against the threshold value.....	64
Figure 6.3: Face Detection Rate against the threshold value.....	65
Figure 6.4: False Detection Rate against the threshold value.....	66
Figure 6.5: Face Detection Rate against the threshold value.....	67
Figure 6.6: False Detection Rate against the threshold value.....	67
Figure 6.7: Face Detection Rate against the threshold value.....	68
Figure 6.8: False Detection Rate against the threshold value.....	69
Figure 6.9: Face Detection Rate against the threshold value.....	69
Figure 6.10: False Detection Rate against the threshold value.....	70
Figure 6.11: Face Detection Rate against the threshold value.....	71
Figure 6.12: False Detection Rate against the threshold value.....	72

Figure 6.13: Obtained outputs.....73

List of Tables

Table 6.1: Network used during the experiment section.....	63
Table 6.2: Face Detection and False Detection Rates as the threshold varied.....	63
Table 6.3: Face Detection and False Detection Rates as the threshold varied.....	65
Table 6.4: Face Detection and False Detection Rates as the threshold varied.....	66
Table 6.5: Face Detection and False Detection Rates as the threshold varied.....	68
Table 6.6: Face Detection and False Detection Rates as the threshold varied.....	69
Table 6.7: Face Detection and False Detection Rates as the threshold varied.....	70

Chapter 1

Introduction

Computer Vision aims to enable computers see and interpret the world. In other words Computer Vision is getting data from still image or image sequence describing a scene and making decisions depending on this description. Research on computer vision has been steadily growing in the last 30 years. This research front delivered many valuable implementations industry, security etc.

Since we are visual creatures and have a highly complex vision system vision is not much of challenge for us. On the other hand, images are nothing but a grid of numbers for a computer system, which makes vision, is very difficult task for computers. Another important problem which makes computer vision difficult is, while humans use 3D information for detection and recognition objects, a computer with a single camera (monocular vision) has to derive same information through the use of 2D information.

Despite these problems there have been significant advances in field such as face detection, recognition, facial expression recognition etc. Most applications that related to human face processing need face detection as a preprocessing step. The accuracy of face detection affects the outcomes of such computer vision applications.

Face detection is a built in hard coded operation for humans, however detecting faces in an image is challenging task for computers because of variability of scale, location, orientation, pose, facial expression occlusion and lighting conditions. Face detection approaches can be classified as knowledge-based, feature based, template matching and appearance methods.

In this thesis I represent a neural network based face detection algorithm for stationary, monocular images. Main differences between this system and previous systems which were used neural network for face detection is size (our system is

much more simple than previous systems) of a network and computational complexity (i.e., processing time). The aim of this thesis is to show that face detection can be done via small sized neural network by using the representational power of neural networks.

This thesis organized as follows chapter 2 gives the general information about the face detection problem, chapter 3 gives the general information about the neural network, chapter 4 explains the neural network based face detection system; chapter 5 introduces the “Neural Network Based Face Detection and Image Processing System”.

Chapter 2

Face Detection

Face detection is the necessary part of lots of computer vision applications such as face and facial expression recognition, human computer interaction etc. In other words, accurate face detection must be accomplished in order to realize these computer vision applications.

The goal of face detection is to determine and locate all faces in a given image regardless of their orientation, size, location, poses, facial expression and lighting conditions. These factors make face detection as a challenging problem.

The face detection problem may be defined as determining whether or not there are any faces in the image and, if present, return the locations and extent of each face.

2.1 Challenges in Face Detection

Challenges in face detection are the variability of pose, structural components, facial expressions, occlusion image orientation and lighting conditions. Below I describe these challenges briefly.

- **Pose:** The images of a face vary due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded. Figure 2.1 is an example of different face poses.

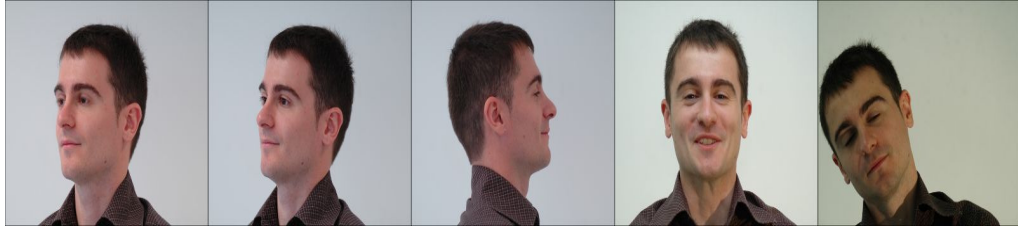


Figure 2.1: Samples of different poses of human face.

- **Presence or absence of structural components:** Facial features such as beards, mustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape, color, and size. Figure 2.2 is an example of presence or absence of structural components.



Figure 2.2: Samples of beards, mustaches and glasses.

- **Facial expression:** The appearance of a face is directly affected by a person's facial expression. Figure 2.3 is an example of different facial expressions.



Figure 2.3: Samples of facial expressions.

- **Occlusion:** Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces. Figure 2.4 is an example of occlusion.

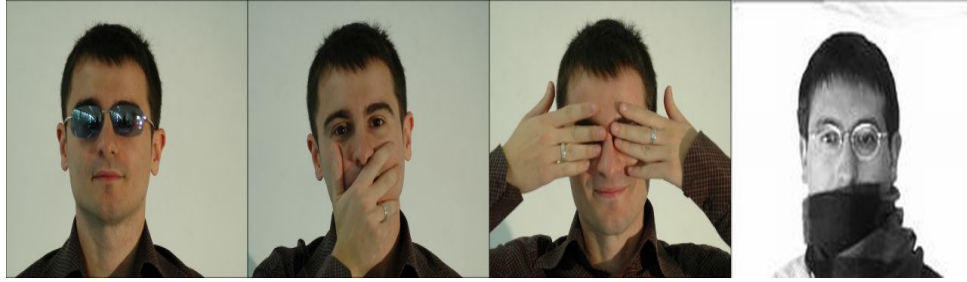


Figure 2.4 Samples of occlusion.

- **Image orientation:** Face images directly vary for different rotations around the optical axis of the camera. Figure 2.5 is an example of an image orientation problem.



Figure 2.5: Image orientation.

- **Imaging conditions:** When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face. Resolution is another factor effecting face detection. Figure 2.6 is an example of an illumination.

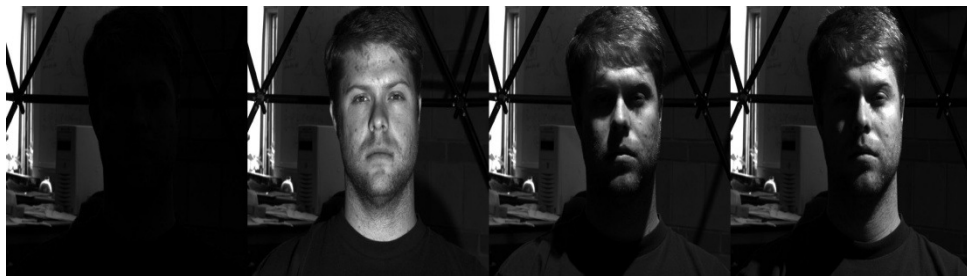


Figure 2.6: Illumination variations.

2.2 Face Detection Methods

In this section I will summarize the state-of-the-art face detection techniques by means of a face detection survey which was published by Yang, Kriegman and Ahuja *et al* [1]. Face detection techniques can be classified as follows.

- **Knowledge – Based Top Down methods:** These rule-based methods use the structural relationship between human facial features such as eyes, nose etc. These methods encode human knowledge of what constitutes a typical face.
- **Bottom-Up Feature-Based Methods:** These methods try to find the structural features of human faces even when the pose, viewpoint, or lighting conditions vary.
- **Template Matching Methods:** Different face patterns are stored in order to describe human face.
- **Appearance-based Methods:** Face detection is implemented by learning from face samples.

2.2.1 Knowledge – Based Top down Methods

Researchers work on knowledge-based top down methods derived rules based on human knowledge of faces. Relationship between facial features can be easily derived from the human knowledge of faces. For example, an ordinary face contains two eyes that are symmetric to each other, a nose and a mouth. Relationship between facial features can be extracted for an average face by using the relative distances and positions of these features. Figure 2.7 is an example of a typical face used in knowledge based top down methods.

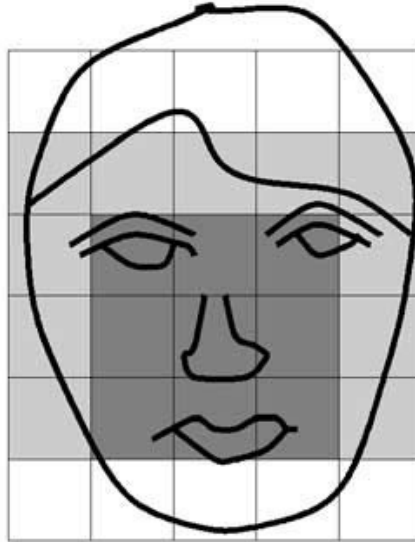


Figure 2.7: Typical face used in knowledge-based top-down methods [1].

First step of a knowledge-based top down method is extracting facial features, the next step is identifying the face candidates, and the final step is applying the verification process in order to reduce false detections.

Main problem of this approach is defining well-defined rules in order to determine human faces. Defined rules should not be too much strict which may fail face detection. Also defined rules should not be too general which may cause too much false detections.

Another problem is detecting faces in different poses because of challenges in specifying all possible cases. However, these methods work well in detecting frontal faces in plain background.

The most well known hierarchical knowledge-based method is represented by Yang and Huang *et al* [2]. Their system consists of three levels of rules. At the highest level all face candidates are found by using window scanning method. The rules at this step are general description of what a face generally looks like. And the rules at the lower levels are related to details of facial features. Another important point in their research is, using multi resolution hierarchy for face detection. At the first step with the lowest resolution shown in Figure 2.8 d, face candidates are determined by the algorithm using the general shape of a face. At the second level, local histogram equalization is applied at the face candidates received from the previous level. At the final level set of rules related to facial features are applied.

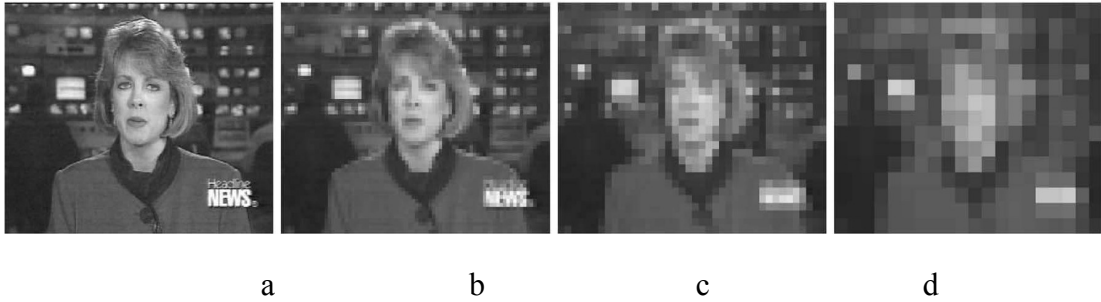


Figure 2.8: Multi resolution hierarchy, a = original image, b,c and d low level resolution images [2].

2.2.2 Bottom-Up Feature-Based Methods

In knowledge based top-down approaches, derived rules are affected by pose, lighting conditions, occlusions etc. Therefore, researchers attempted to find stable rules for facial features which are not affected by lighting conditions, occlusions or pose.

Since humans can detect faces effortlessly in different pose and lighting conditions, one would be inclined to think that, there should be features or properties which are invariant over these variables.

There are lots of methods which firstly detect invariant facial features and then infer the presence of a face. By using edge detectors facial features such as eyebrows, eyes, nose and mouth can be extracted. Then a statistical model is built to describe a relationship between them in order to verify the existence of a face.

One problem with these method is images may be corrupted because of lots of factors such as illumination, occlusion etc.

2.2.2.1 Facial Features

There are many facial features based face detection algorithms and implementations in the literature. Below are the selected facial feature based face detection algorithms.

Localization method for segmentation of faces from a complex background for face detection is proposed by Sirohey *et al* [3]. In proposed algorithm canny edge

detector was used in order to detect edges as shown in Figure 2.9. After edge detection operation a heuristic was applied in order to get edges related to face shape. Then an ellipse is fit to the boundary between the head region and the background. This algorithm achieves 80 percent accuracy on a database of 48 images with complex backgrounds.

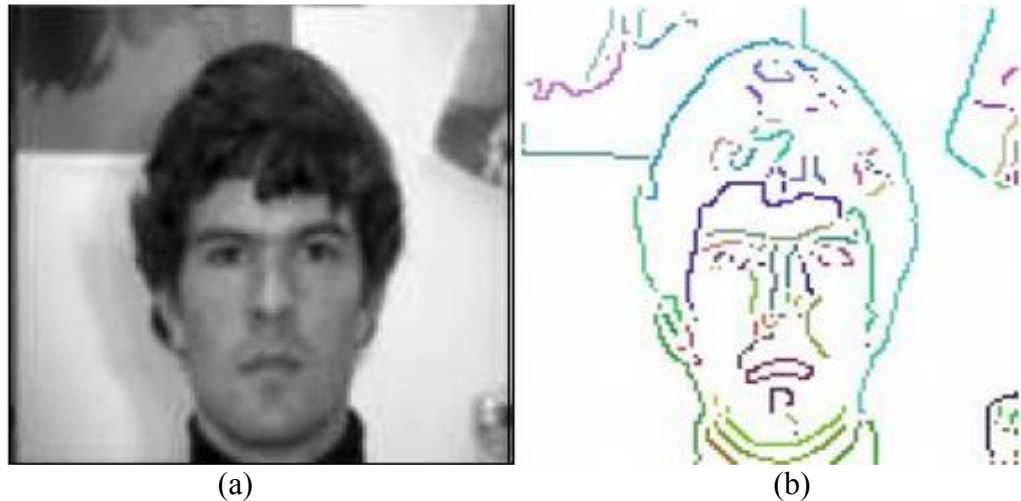


Figure 2.9: Intensity image (a) and edge map from Canny's edge detector (b) [3].

In order to locate facial features, band pass filtering and morphological operations are applied to gray scale images to enhance regions with intensity by Graf *et al* [4]. Typically, there is a significant peak in processed image. Using this peak value and its width, adaptive threshold values are selected in order to generate two binarized images. Connected components are identified in both binarized images to identify the areas of candidate facial features. Final step of the proposed algorithm is, combination of facial features and applying a classifier in order to establish successful face detection.

Another facial features based face detection algorithm was proposed by Han *et al* [5]. Since eyebrows and eyes are the most stable features of human face, they thought that they must be the most important facial features for face detection. First step of the proposed algorithm is applying morphological operators such as closing, clipped difference and thresholding in order to get the pixels at which the intensity values change significantly. Then a labeling process is applied in order to generate the eye- resembling segments. Final step of the algorithm analyses these segments through the geometrical relationships between these facial features.

2.2.2.2 Texture

Since human faces have a distinct texture, it can be used for differentiation of faces from nonface objects. An algorithm related to texture and face detection is proposed by Augusteijn and Skufca *et al* [6]. In their proposed algorithm first step is computing the texture using second order statistical features on subimages of 16X16 pixels. They used a cascade correlation neural network for classification of textures. To detect faces they suggest using votes of the occurrence of hair and skin textures. However they did not report the results of the face detection process.

2.2.2.3 Skin Color

Although human skin color differs from person to person, and race to race, it has been used by many face detection applications. Main reason of why human skin color can be used by these applications is the difference lies largely between their intensity rather than their chrominance. Several color space such as RGB, normalized RGB, HSV, HSI, VCrCb, YIQ, YES and CIE etc. are used for face detection.

Color information can be an efficient tool if the skin color model can be properly adapted for different lighting conditions. Most of the skin color models fail if the lighting conditions are not stable.

One of the applications of human skin color is proposed by Seow, Valaparla and Asari *et al* [7]. They proposed a method for face detection that eliminates the limitations pertaining to the skin color variations among people. They proposed a skin color model in the three dimensional RGB space. They reported that their system can detect faces in different lighting conditions. Figure 2.10 shows the input image and its corresponding result image.



Figure 2.10: (a) Test Image, (b) Result Image [7].

2.2.2.4 Multiple Features

Recently, combining multiple facial features for face detection has become popular. These methods use global features such as skin color and shape for detecting face candidates. Then, these methods use detailed features such as eyebrows, nose and hair for detecting faces. In these methods, skin like pixels are grouped together in order to find the elliptic oval shapes which may be a face region. Finally local features used for verification of face detection.

Using shape and color for face detection is proposed by Sobottka and Pitas *et al* [8]. First step of this proposed algorithm is color segmentation in HSV space for detecting skin-like regions. Next step is determining the connected components for on each skin-like region. For each connected component the best fit ellipse is computed using geometric moments. Then these elliptic regions are determined as face candidates. Figure 2.11 is an example of this algorithm. Then facial features are searched on these regions in order to verify face detection. Their experiment shows a detection rate of 85 percent based on a test set of 100 images.

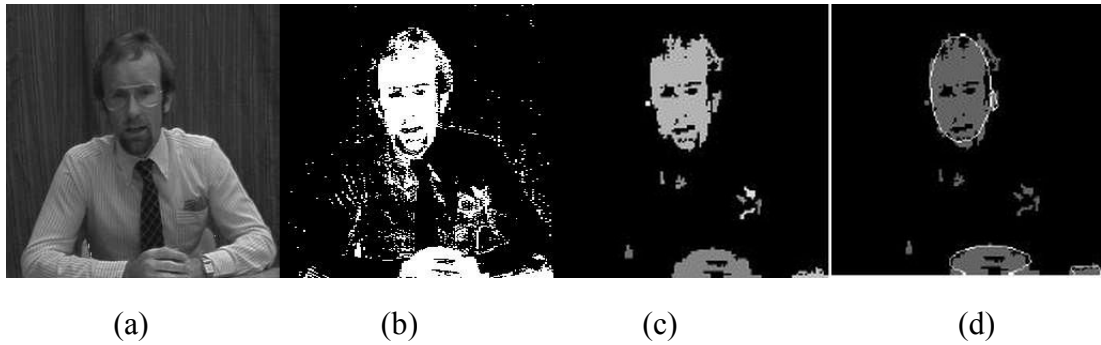


Figure 2.11: (a) original color image, (b) skin segmentation, (c) connected components, (d) best-fit ellipses [8].

The symmetry of facial features has also been used for face detection. Skin-nonskin classification is done in YES color space by means of the class-conditional density function. Then an elliptic face template is used to determine the similarity of the skin color regions. At the last step the eye centers are localized using several cost functions, and the nose and the mouth centers are localized using the distance from the eye centers. This method works well in the single frontal-view face and when both eyes are visible.

2.2.3 Template Matching

In the template matching method, a standard face is manually predefined or parameterized by a function. Correlation values for eyes, nose and mouth are computed independently. The verification of face region is done by correlation values between these features. This approach is easy to implement. However it does not deal with the variation of scale, pose and shape. In order to deal with these problems, there have been new proposed methods such as multiresolution, multiscale and deformable templates.

2.2.3.1 Predefined Templates

An early attempt for face detection is reported by Sakai *et al* [9]. There are many subtemplates for nose, mouth and eyes in order to create a generic model for a face. At the first step, to detect face candidates, correlation between subtemplates and given image is calculated. Then other subtemplates are applied to these face candidate regions to detect faces.

Another algorithm that uses two steps for face detection is presented by Govindarafu *et al* [10]. First of all, face model is extracted based on edges. These extracted edges are used for defining left side, the hair-line and the right side of the frontal face. To obtain edge map of a given image the Marr-Hildreth edge operator is applied. Then a filter is applied to remove edges that are not related to face. Corners are detected to segment contour into feature curves. These features are then labeled as their geometrical shapes and distances. The ratios of the feature pairs performing an edge are compared with the golden ratio and cost is assigned to the edge. If the cost of a group of three feature curves is low, the group becomes a hypothesis. Their system reports a detection rate of approximately 70 percent based on a test set of 50 photographs.

2.2.3.2 Deformable Templates

In order to model facial features deformable templates were used by Yuille *et al* [11]. In this proposed algorithm facial features are described by parameterized templates. Energy function is used for linking edges, peaks and valleys in the input image and corresponding template. Their algorithm shows good performance on tracking the grid features.

Another implementation related to deformable templates which uses both face and intensity information for face representation is proposed by Lanitis *et al* [12]. At the training part of the proposed algorithm sampled contours such as the eye boundary and nose are manually labeled and a vector of sample points is used to represent shape. They used point distribution model to characterize the shape vectors over an ensemble of individuals. A face shape point distribution model can be used to locate face in new images by using active shape models search to estimate the face location. Then the face region is deformed to the average face shape and intensity parameters are extracted. The shape and intensity parameters are used for face detection.

2.2.4 Appearance-Based Methods

Templates in appearance-based methods are not defined by experts. They are learned from examples in images. Techniques used in appearance based methods are related to statistical analysis and machine learning. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection.

Many appearance based methods can be understood in a probabilistic framework. An image can be viewed as random variable x , and this random variable is characterized for faces and nonfaces by the class conditional density functions ($p(x|\text{face})$ and $p(x|\text{nonface})$). Classification of face and nonface objects can be done by Bayesian classifier or maximum likelihood. Since x is high dimensional, $p(x|\text{face})$ and $p(x|\text{nonface})$ are multimodal and it is not yet understood if there are natural parameterized forms for $p(x|\text{face})$ and $p(x|\text{nonface})$ classes, straight forward implementation of Bayesian classifier is impossible.

Another approach in appearance based methods is finding a discriminant function between face and nonface samples. Also neural networks, support vector machines and other kernel methods are proposed.

2.2.4.1 Eigenfaces

Eigenfaces are firstly demonstrated in the research of Kohonen *et al* [13]. In his research a simple neural network is demonstrated to perform face detection for aligned and normalized face images.

Images of faces can be linearly encoded using modest number of basis images as demonstrated by Kirby and Sirovich *et al* [14]. Karhunen-Loeve transform which is also known as Principal Component Analysis is based on this derivation. Given a collection of n by m pixel training images represented as a vector of size $m \times n$, basis vectors spanning an optimal subspace are determined such that the mean square error between the projection of the training images onto this subspace and the original image is minimized. Experiments with a set of 100 images show that a face image of

91×50 pixels can be effectively encoded using only 50 eigenpictures. Figure 2.12 is an example of eigenfaces.

Many works on face detection, recognition and feature extractions such as Turk and Pentland *et al* [15] have adopted the idea of eigenvector decomposition and clustering.



Figure 2.12: Eigenfaces.

2.2.4.2 Distribution-Based Methods

Distribution based system for face detection which demonstrated how the distributions of image patterns from one object class can be learned from positive and negative examples of that class is demonstrated by Sung and Poggio *et al* [16]. The proposed system consists of two parts, distribution-based models for face nonface patterns and a multilayer perceptron classifier. First of all, all training samples are normalized and processed to a 19×19 pixel image and treated as a 361 dimensional pattern. Then patterns are grouped into six face and nonface clusters using a modified k-means algorithm. Each cluster is represented as a multidimensional Gaussian function with a mean image and a covariance matrix. They used two distance matrices. First one is the normalized Mahalanobis distance between the test pattern and the cluster centroid, and the second one is the Euclidean distance between the test pattern and its projection. The last step is the proposed algorithm is multilayer perceptron network to discriminate face windows from nonface windows.

Recently, the Fisherface method which is another implementation of distribution based methods, based on linear discriminant analysis have been shown to outperform the widely used Eigenface method in face detection on several data sets, including the Yale face database where face images are taken under varying lighting conditions.

2.2.4.3 Neural Networks

Neural networks are being successfully applied across an extraordinary range of problems such as optical character recognition, object recognition, and autonomous robot driving. Many neural network architectures have been proposed for the face detection problem due to its suitability as a two class pattern recognition problem. The advantage of using neural networks for face detection problem is the applicability of training a system to detect face and nonface classes. However, the disadvantage that the network architecture is to be widely tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional, and sometimes acceptable performance.

Agui *et al* [17] presented a hierarchical neural network in his research. The first stage of the proposed algorithm consists of two parallel subnetworks. Inputs are intensity values from an original image and intensity values from filtered image. The inputs to the second stage network consist of the outputs from the subnetworks and extracted feature values. An output value at the second stage indicates the presence of a face in the input region. Experimental results show that this method is able to detect faces if all faces in the test images have the same size.

Another method which is developed by Propp and Samal is proposed as neural network based face detection system [18]. Their proposed network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

Another similar hierarchical neural network for face detection is later proposed by Soulie *et al.* [19]. In the proposed neural network input image is scanned with a time-delay neural network in order to detect faces. To cope with size variation, the

input image is decomposed using wavelet transforms. They reported a false negative rate of 2.7 percent and false positive rate of 0.5 percent from a test of 120 images.

The most significant work among all face detection methods which is using neural network for face detection is done by Rowley *et al* [20]. A multilayer neural network is used to learn the face and nonface classes from face/nonface images. The first component of this proposed method is a neural network that receives a 20×20 pixel region of an image and outputs ranging from -1 to 1. -1 for nonface samples and 1 for face samples. They used scan window method for detecting faces in all locations. Also faces which are larger than 20×20 pixels can be detected by resizing the image and applying neural network at all locations. Nearly 1,050 face samples of various sizes, orientations, positions, and intensities are used to train the network. The second component of this method is to merge overlapping detection and arbitrate between the outputs of multiple networks. Simple arbitration schemes such as logic operators (AND/OR) and voting are used to improve performance. Figure 2.13 is the architecture of this proposed algorithm

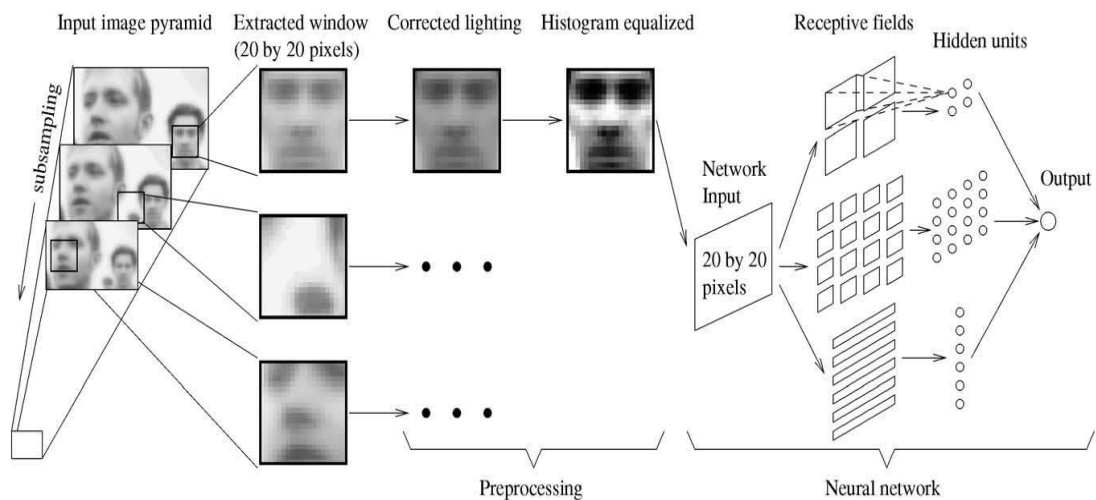


Figure 2.13: Architecture of Rowley *et al* [20].

2.2.4.4 Support Vector Machines

The first application of Support Vector Machines for face detection is presented by Osuna *et al* [13]. Support Vector Machines can be considered as a new paradigm to train polynomial function, neural networks, or radial basis function

classifiers. Support Vector Machines try to minimize an upper bound on the expected generalization error whereas neural networks target to minimize the training error. A Support Vector Machine classifier is a linear classifier where the separating hyperplane is chosen to minimize the expected classification error of the unseen test patterns. This optimal hyperplane is defined by a weighted combination of a small subset of the training vectors, called support vectors. Estimating the optimal hyperplane is equivalent to solving a linearly constrained quadratic programming problem. However, the computation is both time and memory intensive.

An efficient method for Support Vector Machines and face detection is proposed by Osuna *et al* [21]. Osuna developed an efficient way to train an Support Vector Machines and applied it to face detection. Based on two test sets of 10,000,000 test patterns of 19×19 pixels, the proposed system has lower error rates

2.2.4.5 Sparse Network of Winnows

In order to detect faces with different features and expressions, in different poses, and under different lighting conditions a new method that uses SNoW learning architecture is proposed by Yang *et al* [22]. SNoW (Sparse Network of Winnows) is a sparse network of linear functions that utilizes the Winnow update rule . It is specifically tailored for learning in domains in which the potential number of features taking part in decisions is very large, but may be unknown a priori. Some of the characteristics of this learning architecture are its sparsely connected units, the allocation of features and links in a data driven way, the decision mechanism, and the utilization of an efficient update rule.

In training the SNoW-based face detector, 1,681 face images from Olivetti, UMIST , Harvard , Yale , and FERET databases are used to capture the variations in face patterns. To compare with other methods, they report results with two readily available data sets which contain 225 images with 619 faces . With an error rate of 5.9 percent, this technique performs as well as or, better than the other methods that are evaluated on the data set.

2.2.4.6 Naïve Bayes Classifier

Schneiderman and Kanade *et al* proposed a method that estimates the Joint probability of local features at multiple resolutions using a naive Bayes classifier[23]. They emphasize local features because some local features of a face are more unique than others; e.g. the intensity patterns around the eyes are much more distinctive than the pattern found around the cheeks. First reason why they used naive bayes classifier is that, it provides better estimation of the conditional density functions of these subregions. Furthermore, a naive Bayes classifier provides a functional form of the posterior probability to capture the joint statistics of local appearance and position on the face. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. Under this formulation, their method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities

Another method related to joint statistical models of local features was developed by Rickert *et al.* [24]. In order to extract local features multiscale and multiresolution filters are applied to the input image. The distribution of the features vectors is estimated by clustering the data and then forming a mixture of Gaussians. After the model is learned and further refined, test images are classified by computing the likelihood of their feature vectors with respect to the model. Their experimental results on face and car detection show interesting and good results

2.2.4.7 Hidden Markov Model

Hidden Markov Model (HMM) is based on the following assumption. Patterns can be characterized as a parametric random process and that the parameters of this process can be estimated in a perfect, well-defined manner. First step of using HMM for face detection problem is forming a HMM model. In order to form a HMM model, number of hidden states need to be decided .Then, HMM can be trained in order to learn the transitional probability between states from the examples where each example is represented as a sequence of observations. After the HMM has been

trained, the output probability of an observation determines the class to which it belongs.

A face region can be divided into several subregions such as the forehead, eyes, nose, mouth, and chin. A face region can then be recognized by a process in which these subregions are observed in an appropriate order (from top to bottom and left to right). Instead of relying on accurate alignment as in template matching or appearance based methods, this approach aims to associate facial subregions with the states of a continuous density

During training and testing, an image is scanned in some order (usually from top to bottom) and an observation is taken as a block of pixels. For face regions, the boundaries between strips of pixels are represented by probabilistic transitions between states, and the image data within a region is modeled by a multivariate Gaussian distribution. An observation sequence consists of all intensity values from each block. The output states correspond to the classes to which the observations belong. After the HMM has been trained, the output probability of an observation determines the class to which it belongs.

One state is responsible for characterizing the observation vectors of human foreheads, and another state is responsible for characterizing the observation vectors of human eyes. For face localization, an HMM is trained for a generic model of human faces from a large collection of face images. If the face likelihood obtained for each rectangular pattern in the image is above a threshold, a face is located.

Chapter 3

Neural Networks

3.1 Introduction to Neural Networks

Artificial Intelligence is one of the topics of Computer Science that attempts to give computers a humanlike intelligence. Inspired from the human brain Neural Networks (NN) are one of the most important tools which enable computers to make decisions as humans do. The human brain consists of a network of over a billion interconnected neurons. Neurons are individual cells that can process small amounts of information and then activate other neurons to continue the process.

Essentially a NN is a simplified model of human brain. A NN transforms inputs into outputs to the best of its ability.

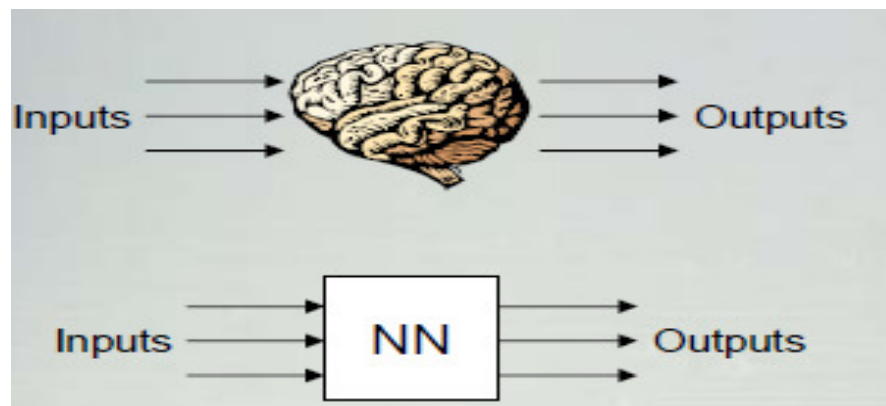


Figure 3.1: Neural network transforms inputs into outputs [25].

Neural networks are being successfully applied to variety of problem domains such as computer science, finance, medicine, engineering, geology and physics. Also, anywhere that there are problems of prediction, classification or control, NNs

can be considered as a candidate solution. The success of neural networks is dependent on the following factors:

- Power. Since neural networks are very advanced modeling techniques, they are capable of modeling extremely complex functions. In particular, neural networks are nonlinear. For many years linear modeling has been the commonly used technique in most of the problem domains. Where the linear approximation was not valid the proposed models produced unsuccessful results. Neural Networks are capable of simulating nonlinear problems.
- Ease of use. Neural Networks learn by example. A neural network user gathers representative data, and then invokes training algorithm to automatically learn the structure of the data. The user does need to have some heuristic knowledge of how to select and prepare data, how to select an appropriate neural network, and how to interpret the results; the neural network can derive the discriminating features from the data. In other words, the level of user knowledge needed to successfully apply neural networks is much lower than would be the case using some more traditional nonlinear statistical methods.

3.2 Biological Neuron

To construct “human like thought” computer researchers try to simulate the human brain. Since human brain is so complex, researcher avoid considering the human brain as a whole. Individual cells that make up the human brain are studied.

Human brain is composed of neuron cells. A neuron cell, as seen in Figure 3.2 is the basic building block of the human brain. A neuron accepts signals from the dendrites. When a neuron accepts a signal, that neuron may fire. When a neuron fires, a signal is transmitted over the neuron's axon. Ultimately the signal will leave the neuron as it travels to the axon terminals. The signal is then transmitted to other neurons or nerves.

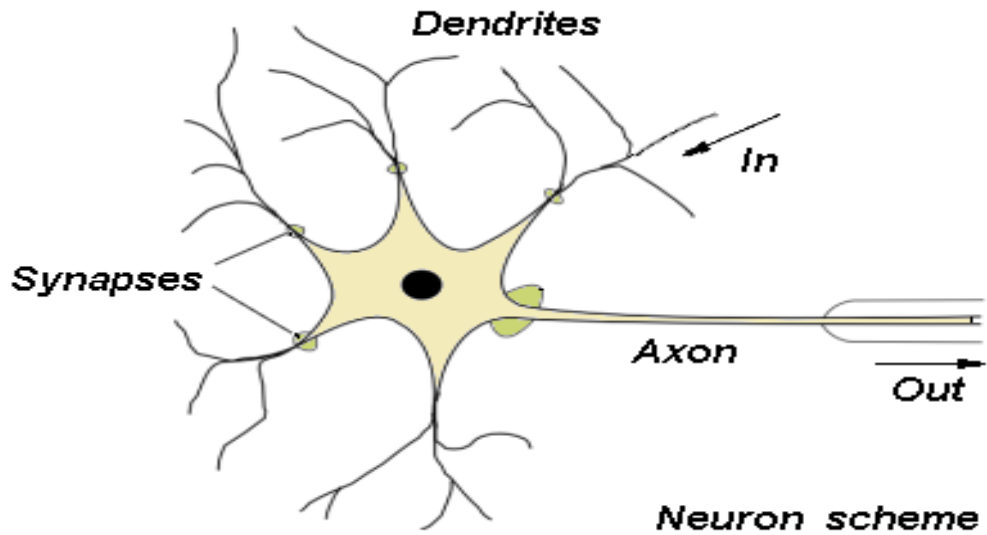


Figure 3.2: A Neuron Cell [25].

Neurons transmitted the analog signal. However, modern computers are digital machines, they require a digital signal. A digital computer processes information as either on or off. This is the basis of the binary digits zero and one. The presence of an electric signal represents a value of one, whereas the absence of an electrical signal represents a value of zero. Figure 3.3 shows a representative digital signal

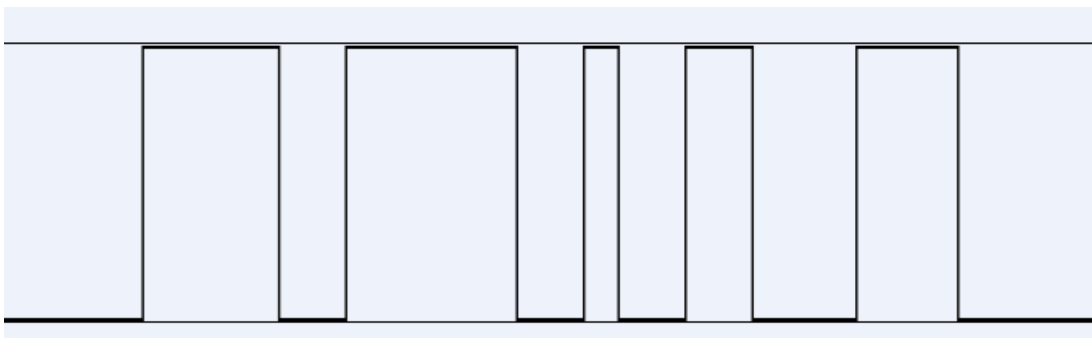


Figure 3.3: A Digital Signal [25].

Biological neural networks are analog. Simulating analog neural networks on a digital computer can present some challenges. Neurons accept an analog signal

through their dendrites. Because this signal is analog the voltage of this signal will vary. If the voltage is within a certain range, the neuron will fire. When a neuron fires, a new analog signal is transmitted from the firing neuron to other neurons. This signal is conducted over the firing neuron's axon and fed to other neurons through synapses. Synapses are the input/output interfaces of neurons.

Firing or not firing a neuron corresponds to making a decision. These are extremely low level decisions. It takes firing of much higher numbers of such neurons to read this sentence. Higher level decisions are the result of the collective input and output of many neurons. Biological neurons are capable of making basic decisions. This model is what artificial neural networks are based on.

3.3 Simulating the Biological Neuron

A computer can be used to simulate a biological neural network. This computer simulated neural network is called an artificial neural network. Artificial neural networks are almost always referred to simply as neural networks.

In order to simulate biological neural systems, an artificial neuron is defined as follows:

It receives a number of inputs (either from original data, or from the output of other neurons in the neural network). Each input comes via a connection that has a weight; these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is calculated, and the threshold subtracted. Then the activation of the neuron is composed.

The activation signal is passed through an activation function (transfer function) to produce the output of the neuron.

If the step activation function is used then the neuron acts just like the biological neuron. Actually, the step function is rarely used in artificial neural networks. Also weights can be negative or positive,

If a neural network is used, there must be inputs and outputs. Inputs and outputs correspond to sensory and motor nerves such as those coming from the eyes

and leading to the hands. However, there also can be hidden neurons that play an internal role in the network. The input, hidden and output neurons need to be connected together.

A simple network has a feedforward structure: signals flow from inputs, forwards through any hidden units, in order to reach the output units. Such a structure has stable behavior. However, if the network is recurrent it can be unstable, and has very complex dynamics. Recurrent networks are very interesting to researchers in neural networks, but so far it is the feedforward structures that have proved most useful in solving real problems.

A typical feedforward network has neurons arranged in a distinct layered topology. The input layer is not really neural at all: these units simply serve to introduce the values of the input variables. The hidden and output layer neurons are each connected to all of the units in the preceding layer. Again, it is possible to define networks that are partially-connected to only some units in the preceding layer; however, for most applications fully-connected networks are better.

When the network is executed, the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer, and subtracting the threshold. The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network.

3.4 History of Neural Networks

Although human brain have been working for thousands of years, only with the advent of modern electronics that people have started to try for the human brain and the processes of thinking. The modern period of neural network research is started with the work done by neuro-physiologist, Warren McCulloch and Walter Pitts in 1943. They wrote a paper on how neurons might work, and they designed and built a primitive artificial neural network using simple electric circuits.

The book "The Organization of Behavior" which is written by Donald Hebb in 1949 is the next major development in neural network system the book supported and further reinforced McCulloch-Pitts's theory about neurons and how they work. A major point brought forward in the book described how neural pathways are strengthened each time they were used.

Since traditional computing which slow down the development of neural Networks began in 1950s. However certain individuals continued research into neural networks. In 1954 Marvin Minsky wrote a doctorate thesis, "Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem", which was concerned with the research into neural networks. He also published a scientific paper entitled, "Steps Towards Artificial Intelligence" which was one of the first papers to discuss AI in detail. In 1956 the Dartmouth Summer Research Project on Artificial Intelligence began researching AI, what was to be the primitive beginnings of neural network research.

A neuro-biologist at Cornell University who is called Frank Rosenblatt began working on the Perceptron in 1958. The perceptron was the first "practical" artificial neural network. One major downfall of the perceptron was that it had limited capabilities and this was proven by Marvin Minsky and Seymour Papert's book of 1969 entitled, "Perceptrons".

ADALINE (ADaptive LINear Elements) and MADELINE (Multiple ADaptive LINear Elements) models are developed by the researchers Bernard Wildrow and Marcian Hoff between 1959 and 1960. These were the first neural networks that could be applied to real problems.

In 1982 John Hopfield of Caltech presented a paper to the scientific community in which he stated that the approach to AI should not be to purely imitate the human brain but instead to use its concepts to build machines that could solve dynamic problems. He showed what such networks were capable of and how they would work.

Rumelhart, Hinton and Williams reported back on the developments of the back-propagation algorithm in 1986. The paper discussed how back-propagation

learning had emerged as the most popular learning set for the training of multi-layer perceptrons.

With the dawn of the 1990's and the technological era, many advances into the research and development of artificial neural networks are occurring all over the world. Nature itself is living proof that neural networks do in actual fact work. The challenge today lies in finding ways to electronically implement the principals of neural network technology. Electronics companies are working on three types of neuro-chips namely, digital, analog, and optical. With the prospect that these chips may be implemented in neural network design, the future of neural network technology looks very promising.

3.5 Neural Network Types

There are many different kinds of neural networks. The most common used are single-layer feedforward, multilayer, and recurrent network.

3.5.1 Single-Layer Feedforward Neural Network

Single-layer feedforward neural network is the earliest kind of neural network. It consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. It is the simplest kind of feed-forward network.

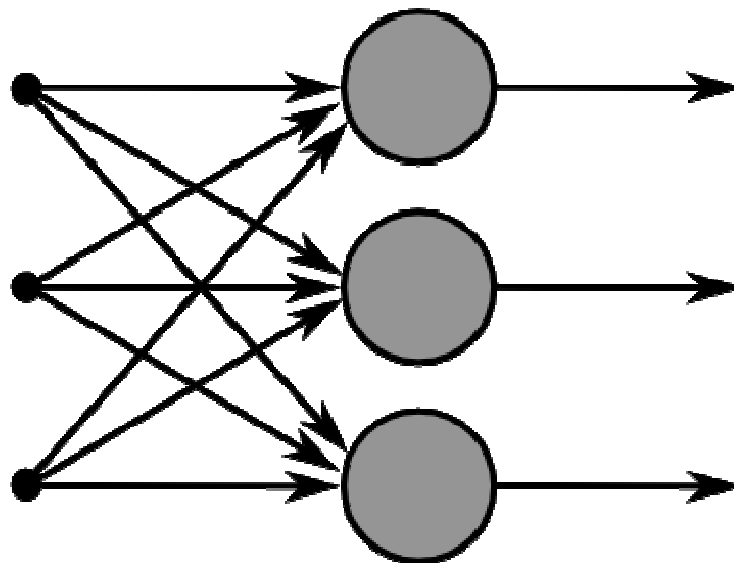


Figure 3.4: Single layer feed forward neural network

3.5.2 Multilayer Neural Network

Multilayer Neural Network consists of multiple layers of computational units. They are interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer.

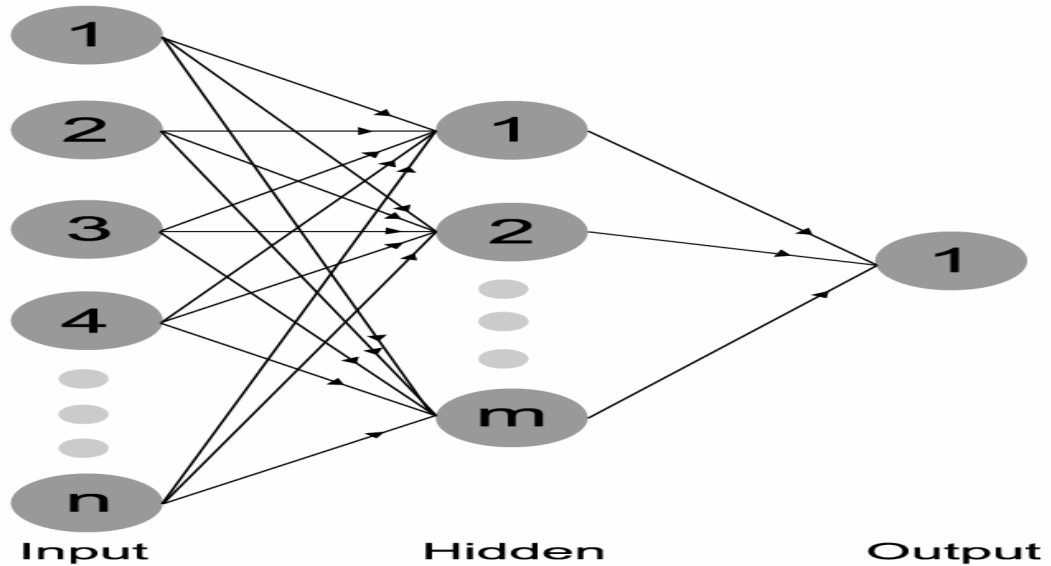


Figure 3.5: Multilayer feedforward neural network

3.5.3. Recurrent Neural Network

A recurrent neural network contains connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior.

Recurrent neural networks must be approached differently from feedforward neural networks, both when analyzing their behavior and training them. Recurrent neural networks can also behave chaotically. Usually, dynamical systems theory is used to model and analyze them. While a feedforward network propagates data linearly from input to output, recurrent networks also propagate data from later processing stages to earlier stages

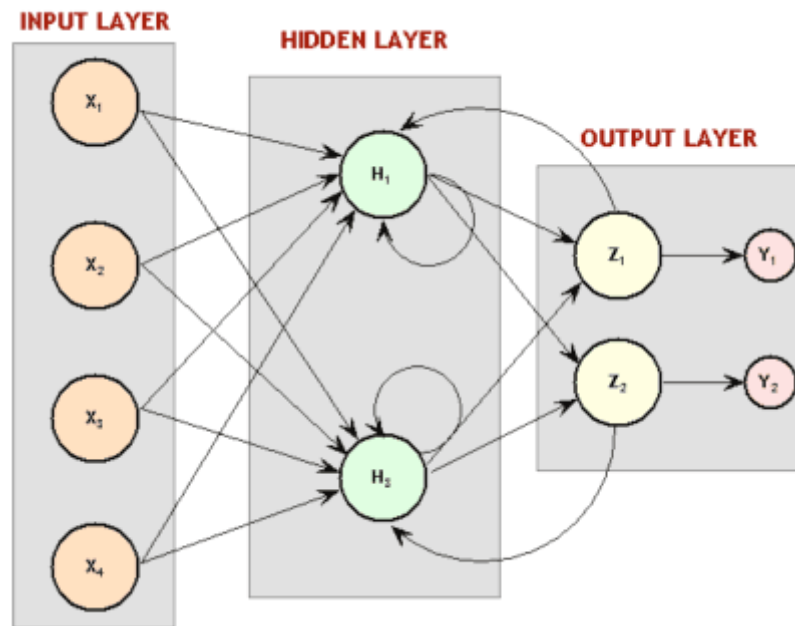


Figure 3.6: Recurrent Neural Network

3.6 Solving Problem with Neural Networks

Most important thing for most computer technologies is when to use the technology and when not to. Neural Networks are no different. In order to apply neural Networks technology to problem neural network user should knows what neural network structure, which is most applicable to a given problem.

3.6.1 Problems Can Not Be Solved by Neural Network

Most programs do not require a neural network. If a program which is easily written out as a flowchart, does not need a neural network. If a program consists of well defined steps, normal programming techniques will enough.

Since neural networks can learn, program which has an unchanged logic will not suitable to use neural networks.

Finally, neural networks are often not suitable for problems where someone knows exactly how the solution was derived. A neural network can become very adept at solving the problem for which the neural network was trained. But the neural network cannot explain its reasoning. The neural network knows because it was

trained to know. The neural network cannot explain how it followed a series of steps to derive the answer.

3.6.2 Problems Can Be Solved by Neural Network

Although there are many problems that neural networks are not suited towards there are also many problems that a neural network is quite adept at solving. Neural networks can often solve problems with fewer lines of code than a traditional programming algorithm. It is important to understand what these problems are. Neural networks are particularly adept at solving problems that cannot be expressed as a series of steps. Neural networks are particularly useful for recognizing patterns, classification into groups, series prediction and data mining. Pattern recognition is perhaps the most common use for neural networks. The neural network is presented a pattern. This could be an image, a sound, or any other sort of data. The neural network then attempts to determine if the input data matches a pattern that the neural network has memorized.

Classification is a process that is closely related to pattern recognition. A neural network trained for classification is designed to take input samples and classify them into groups. These groups may be fuzzy, without clearly defined boundaries. These groups may also have quite rigid boundaries.

Series prediction uses neural networks to predict future events. The neural network is presented a chronological listing of data that stops at some point. The neural network is expected to learn the trend and predict future values.

3.7 Structure of Neural Network

A neural network is composed of several different elements. Neurons which are the most basic unit are interconnected to each other. These connections are not equal, as each connection has a connection weight. Groups of networks come together to form layers.

3.7.1 The Neuron

The neuron which is the basic building block of the neural network is a communication conduit. The neuron both accepts input and produces output. The neuron receives its input either from other neurons or the user program. Similarly the neuron sends its output to other neurons or the user program.

When a neuron produces output, that neuron is said to activate, or fire. A neuron will activate when the sum of its inputs satisfies the neuron's activation function. Consider a neuron that is connected to k other neurons. The variable w represents the weights between this neuron and the other k neurons. The variable x represents the input to this neuron from each of the other k neurons. Therefore we must calculate the sum of every input x multiplied by the corresponding weight w . This is shown in the following equation

$$u = \sum_k w_k x_k$$

This sum must be given to the neuron's activation function. The net value as expressed by the basis function, u , will be immediately transformed by a nonlinear activation function of the neuron. For example, the most common activation functions are step, ramp, sigmoid, and Gaussian function.

- Sigmoid Function

$$f(u_i) = \frac{1}{1 + e^{-u_i/\sigma}}$$

- Gaussian function

$$f(u_i) = ce^{-u_i^2/\sigma^2}$$

- Hyperbolic Tangent

$$f(u_i) = \frac{e^u + e^{-u}}{e^u - e^{-u}}$$

The hyperbolic tangent threshold method will return values according to Figure 3.7. The TANH activation method will be useful when user needs output greater than and less than zero. If only positive numbers are needed, then the Sigmoid threshold method will be used.

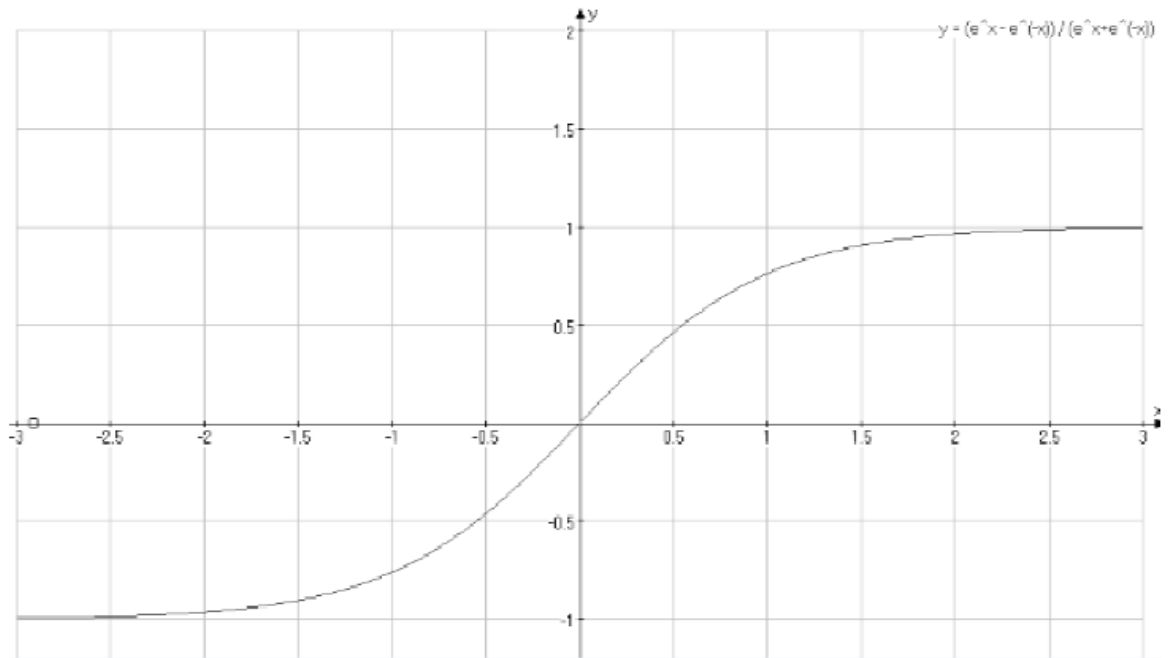


Figure 3.7: TanH activation Function

3.7.2 Neuron Connection Weights

Neurons are usually connected together. These connections are not equal, and can be assigned individual weights. These weights give the neural network the ability to recognize certain patterns. Adjust the weights and the neural network will recognize a different pattern. Adjustment of these weights is a very important operation. The process of training is adjusting the individual weights between each of the individual neurons.

3.7.3 Neuron Layers

Neurons are often grouped into layers. Layers are groups of neurons that perform similar functions. There are three types of layers.

Input Layer: The input layer is the layer of neurons that receive input from the user program. The layer of neurons that send data to the user program is the output layer.

Hidden Layer: The hidden layer is in between the input layer and output layer. Hidden layer neurons are only connected to other neurons and never directly interact with the user program. The input layer presents pattern to the hidden layer. Then the hidden layer presents information on to the output layer. Finally the user program collects the pattern generated by the output layer.

Output Layer: Displays the results of the network to user.

The input and output layers are not just there as interface points. Every neuron in a neural network has the opportunity to affect processing. Processing can occur at any layer in the neural network.

Not every neural network has many layers. The hidden layer is optional. The input and output layers are required, but it is possible to have one layer act as both an input and output layer. Figure 3.8 is an example of Neural Network Layers.

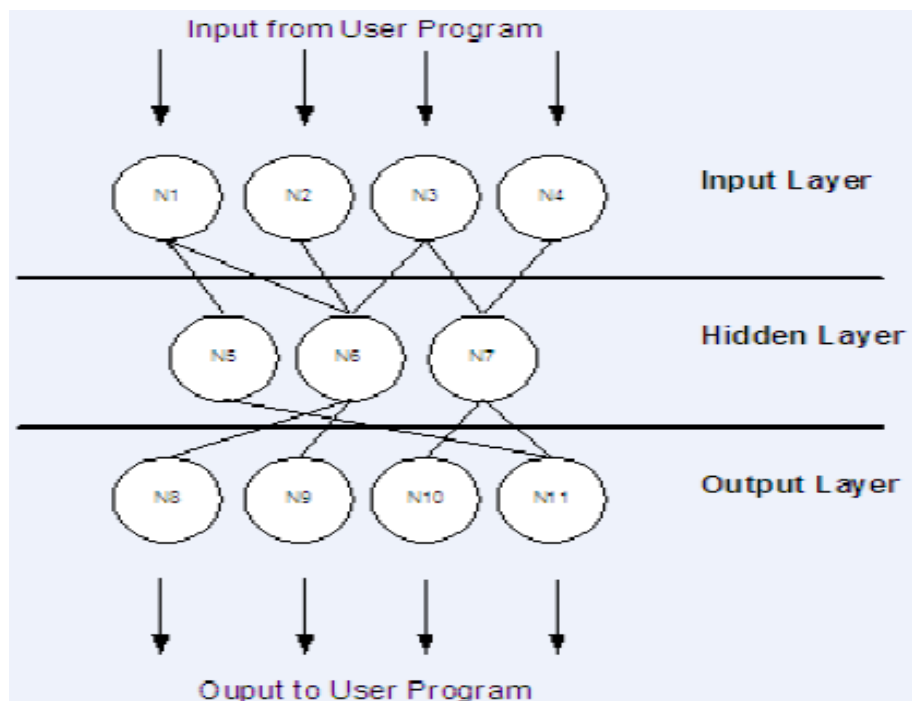


Figure 3.8: Neural Network Layers [25].

3.8 Training the Neural Network

Neural network consists of neurons which are connected to each other via synapses. These connections allow the neurons to signal each other as information is processed. Not all connections are equal. Each connection is assigned a connection weight. These weights are what determine the output of the neural network. Therefore it can be said that the connection weights form the memory of the neural network. Training is the process by which these connection weights are assigned. Most training algorithms begin by assigning random numbers to the weight matrix. Then the validity of the neural network is examined. Next the weights are adjusted based on how valid the neural network performed. This process is repeated until the validation error is within an acceptable limit. There are many ways to train Neural Networks such as supervised, unsupervised and various hybrid approaches.

Supervised training is accomplished by giving the neural network a set of sample data along with the anticipated outputs from each of these samples. Supervised training is the most common form of neural network training. As supervised training proceeds the neural network is taken through several iterations, or epochs, until the actual output of the neural network matches the anticipated output, with a reasonably small error. Figure 3.9 is schematic diagram of supervised learning.

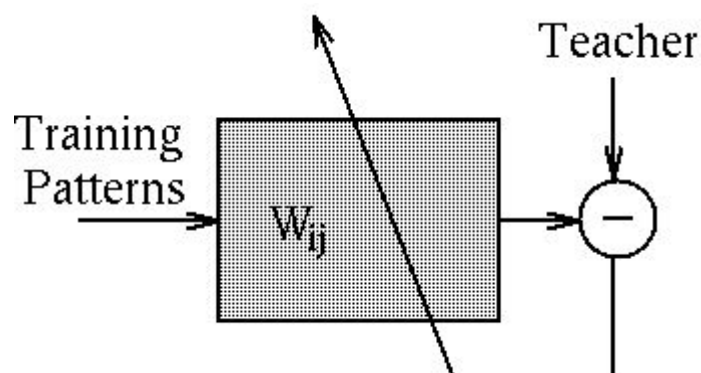


Figure 3.9: Schematic diagram of supervised Learning

Unsupervised training is similar to supervised training except that no anticipated outputs are provided. Unsupervised training usually occurs when the

neural network is to classify the inputs into several groups. The training progresses through many epochs, just as in supervised training. As training progresses the classification groups are “discovered” by the neural network. Figure 3.10 is schematic diagram of unsupervised learning.

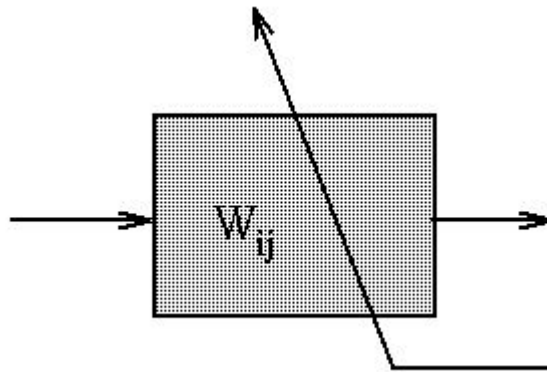


Figure 3.10: Schematic diagram of unsupervised learning

There are several hybrid methods that combine several of the aspects of supervised and unsupervised training. One such method is called reinforcement training. In this method the neural network is provided with sample data that does not contain anticipated outputs, as is done with unsupervised training. However, for each output, the neural network is told whether the output was right or wrong given the input. It is very important to understand how to properly train a neural network.

3.9 Backpropagation Algorithm

The Backpropagation algorithm was first proposed by Paul Werbos in the 1970's. However, it wasn't widely used until rediscovered in 1986 by Rumelhart and McClelland.

Backpropagation is a very common method for training multilayer feed forward networks. Backpropagation can be used with any feed-forward network that uses a threshold function which is differentiable. It is this derivative function that we will use during training.

To train the neural network a method must be determined to calculate the error. As the neural network is trained, the net is presented with samples from the training

set. The result obtained from the neural network is then compared with the anticipated result that is part of the training set. The degree to which the output from the neural network matches this anticipated output is the error.

To train the neural network, we must try to minimize this error. To minimize the error the neuron connection weights and biases must be modified. We must define a function that will calculate the error of the neural network. This error function must be differentiable. Because the network uses a differential threshold function the activations of the output neurons can be thought of as differentiable functions of the input, weights and bias. If the error function is also differentiable error, such as the "sum of square" error function the error function itself is a differentiable function of the these weights. This allows us to evaluate the derivative of the error using the weights. Then using these derivatives we can find weights and bias that will minimize the error function.

There are several ways that weights that minimize the error function can be found. The most popular is by using the gradient descent method. The algorithm that evaluates the derivative of the error function is known as backpropagation, because it propagates the errors backward through the network.

Backpropagation algorithm can be summarized as follow:

- 1- Present a training sample to the neural network.
- 2- Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
- 3- For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
- 4- Adjust the weights of each neuron to lower the local error.
- 5- Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
- 6- Repeat from step 3 on the neurons at the previous level, using each one's "blame" as its error.

3.11 Validating the Neural Network

After training the neural network it must be evaluated to see if it is ready for actual use. This final step is important so that it can be determined if additional training is required. To correctly validate a neural network validation data must be set aside that is completely separate from the training data.

Once the network was properly trained the second group data would be used to validate the neural network. It is very important that a separate group always be maintained for validation. First training a neural network with a given sample set and also using this same set to predict the anticipated error of the neural network a new arbitrary set will surely lead to bad results. The error achieved using the training set will almost always be substantially lower than the error on a new set of sample data. The integrity of the validation data must always be maintained.

This brings up an important question. What exactly does happen if the neural network that is just trained performs poorly on the validation set? If this is the case it could mean that the initial random weights were not good. Rerunning the training with new initial weights could correct this. While an improper set of initial random weights could be the cause, a more likely possibility is that the training data was not properly chosen. If the validation is performing badly this most likely means that there was data present in the validation set that was not available in the training data. The way that this situation should be solved is by trying a different, more random, way of separating the data into training and validation sets.

Chapter 4

Neural Network Based Face Detection

4.1 Background

Face detection is a problem related to face processing and machine learning. In face detection problem space, there are two classes, face and non face, which make face detection a classification problem. Distribution of these classes can be learned via machine learning and statistical methods.

Although structures of faces are similar and facial features are arranged in roughly the same spatial configuration, even among images of the same person's face have significant differences because of illumination, facial expressions, pose etc. Therefore template matching and facial features based face detection methods fail to detect faces.

Since face and nonface spaces are very complex, linear discriminant functions cannot separate these two classes linearly. Face detection problem can be solved by using the power of neural networks because face detection is a nonlinear problem and neural networks can generalize the face space using the training set.

Neural detectors have been proposed recently and the experiments give extremely satisfactory results which mean that neural networks can be used as face detector. In particular, advanced training methods have been used to incorporate the wide distribution of the frontal view of face patterns in neural network knowledge. An efficient solution to the selection problem of non-face images has already been presented aiming at minimizing the false acceptance error. This technique reduces the number of falsely recognized faces in arbitrary images, a critical factor for developing practical face recognizers.

The most successful neural network based face detector was developed by Rowley, Baluja and Kanade *et al* [20]. They presented a neural network-based algorithm to detect upright, frontal views of faces in gray-scale images. The algorithm works by applying one or more neural networks directly to portions of the input image, and arbitrating their results. Each network is trained to output the presence or absence of a face. The algorithms and training methods are designed to be general, with little customization for faces. Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical “nonface” images. They avoid the problem of using a huge training set for nonfaces by selectively adding images to the training set as training progresses. This “bootstrap” method reduces the size of the training set needed. The use of arbitration between multiple networks and heuristics to clean up the results significantly improves the accuracy of the detector. The size of a window used by their system is 20×20 pixels. They found that the system is able to detect 90.5% of the faces over a test set of 130 complex images, with an acceptable number of false positives

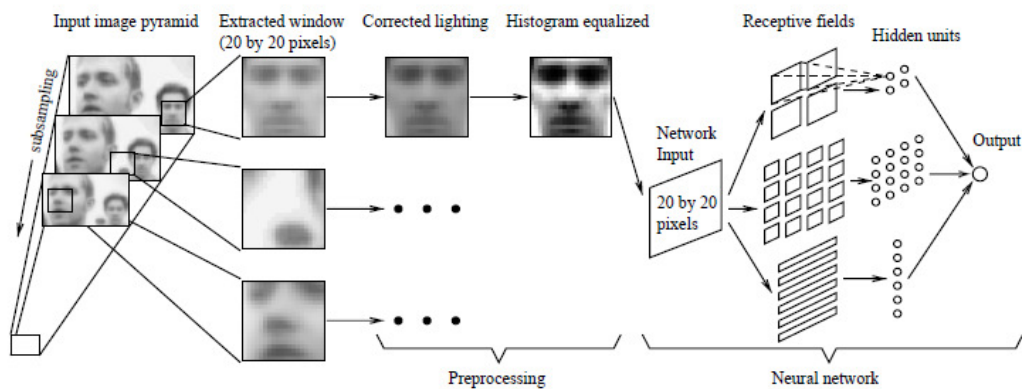


Figure 4.1: Algorithm used by Rowley for face detection [20].

Another implementation of neural networks on face detection problem is done by Anifantis, Dermatos and Kokkinakis *et al* [26]. They presented a neural detector of frontal faces in gray scale images under arbitrary face size, orientation, facial expression, skin color, lighting conditions and background environment. In a two-level process, a window normalization module reduces the variability of the features and a neural classifier generates multiple face position hypotheses. Extended experiments carried out in a test-bed of 6406 face images, have shown that the face

detection accuracy is increased significantly when non-linear and probabilistic illumination equalizers pre-process the sub-images. Moreover, better results can be achieved in case of training the neural detector using positional and orientation normalized face examples. In this case the neural face detector has the capability to locate both position and orientation of a face. The size of a window used by their system is 30×30 pixels. In the multiple face position hypotheses generated by the proposed neural method, 98.3% detection accuracy, the highest reported in the literature, was measured

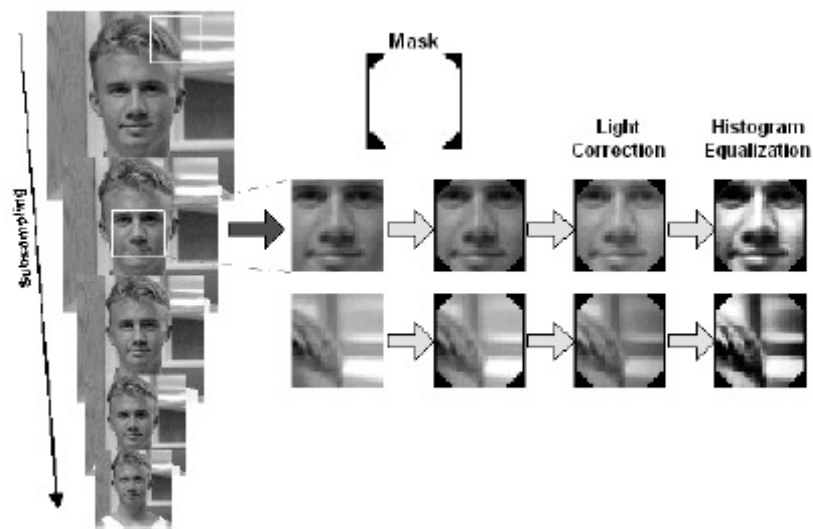


Figure 4.2: Filtering process using by Anifantis for face detection [26].

Another implementation of neural networks for face detection problem is submitted by Garcia and Delakis *et al* into the literature [27]. They presented a connectionist approach for detecting and precisely localizing semi-frontal human faces in complex images, making no assumption about the content or the lighting conditions of the scene, or about the size or the appearance of the faces. They proposed a convolutional neural network architecture designed to recognize strongly variable face patterns directly from pixel images with no preprocessing, by automatically synthesizing its own set of feature extractors from a large training set of faces. The size of a window used by their system is 32×36 pixels. Their system detection rate is changed between 85% and 97% according to variables that are using during the training and running, and datasets.

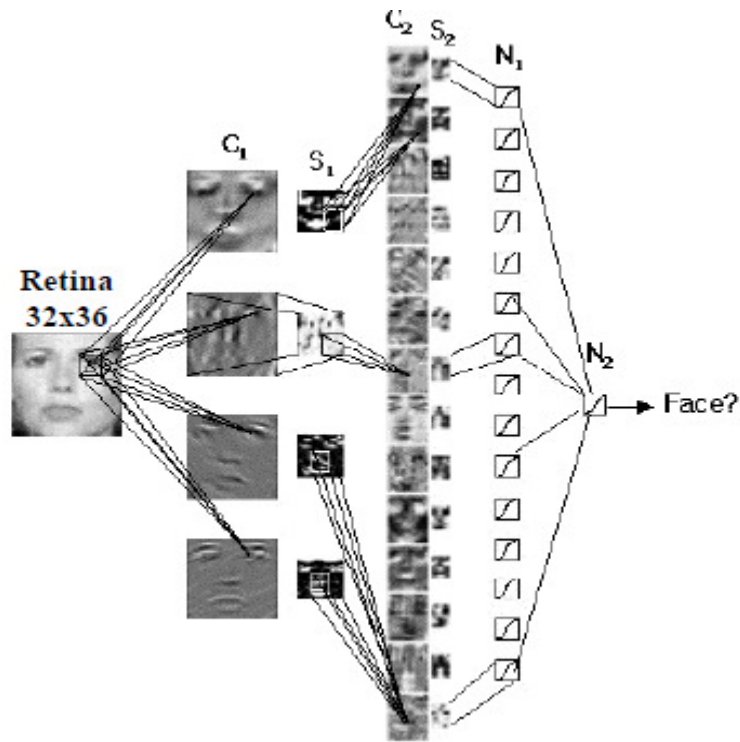


Figure 4.3: Convolutional Neural Network structure used by Garcia and Delakis for face detection [27].

Also neural network based face detector which combined with other face detection methods is proposed by FeÂraud, Bernier, Viallet and Collobert *et al* [28]. The proposed system used skin color information in order to reduce the number of false detections. The size of a window used by their system is 15×20 pixels. Their approach first implements simple processes, based on standard image processing and then more sophisticated processes based on statistical analysis. They described the different components of the face detector are described: a motion filter, a color filter, a prenetwork filter, and a large neural network filter based on a new model of neural network. A combination of neural networks is used to extend the face detection ability in orientation. They presented a fast search algorithm for face detection . It speeds up the detection process by a factor of 25. The reported detection rate in that paper vary between 74% and 87% depends on the database and the model.

4.2 Description of the our System

The purpose of this thesis is to detect faces by using the power of neural networks. We have developed a neural network based face detector which detects upright frontal faces. The proposed system consists of 3 stages. The first step is preprocess step in which image is prepared for the usage of a neural network. In the second step we apply a neural network to all locations of the image with several sizes. The final step is just combining the multiple detections and removing the false detections. In this section these 3 stages will be explained.

4.2.1 Preprocess Step

The first step in our system is preprocessing. The preprocessing operation during training the network has differences from the preprocessing operation during running the network.

- For Training
 - Histogram Equalization is applied to all training samples (both face and nonface samples).
 - All training samples resized to 10×10 pixel size
 - Mask is applied to image in order to ignore the background
 - Pixel values are stored to txt file which will be used by the network during the training.
- For Running
 - Histogram Equalization is applied to given image
 - Image pyramid is built from the histogram equalized image.
 - 10×10 window is applied to all locations of images at all scales
 - Mask is applied to all of 10×10 windows.

Histogram equalization which is the first operation for both running and training is applied at this step. This normalization operation expands the range of brightness intensities in the masked window. This compensates for differences in camera response curves and improves contrast in some cases.

Histogram equalization is a non-linear process reassigning the brightness values of pixels on the basis of the image histogram. Individual pixels retain their brightness order (each pixel remains brighter or darker than other pixels) but the values are shifted, so that an equal number of pixels have each possible brightness value. In many cases, this spreads out the values in areas where different regions meet, showing details in areas with a high brightness gradient. This process is quite simple and consists of four steps:

1. the running sum of the histogram values is estimated,
2. the sum acquired from the previous step is normalized with the total number of pixel
3. the normalized sum is multiplied by the maximum gray level value and rounded,
4. the original gray level values are mapped to the results from Step (3) using one-to-one correspondence

Figure 4.4 is an example of histogram equalization.



(a) (b)
Figure 4.4: (a) Original Image, (b) Histogram equalized image

Second operation for running part of the preprocessing operation is creating the Image pyramid from the histogram equalized image. Image pyramid is used to detect faces in multiscale. Resizing starts with the original image size and then continues up to size of 10×10 pixel size. At each resizing operation image's size is decreased by a factor of 1.2. Figure 4.6 shows the image pyramid of the histogram equalized image.

Next step is applying the oval mask to image in order to ignore the background pixels. Figure 4.5 shows the oval mask that is used by our system.

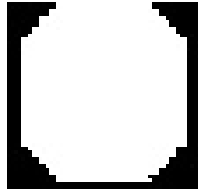


Figure 4.5: Oval mask used by the system

The last step for the training part is, writing pixel values to a text file. Neural network which is developed by us is using pixel values of the image for both training and running. Therefore, in order to use neural network as a classifier pixel values should be presented to a neural network. We have developed a command based system which is called “Neural Network Based Detector” is used for translating image to a txt file. This command based system will be explained in the next section with more details. Figure 4.7 is an example of image scanning.

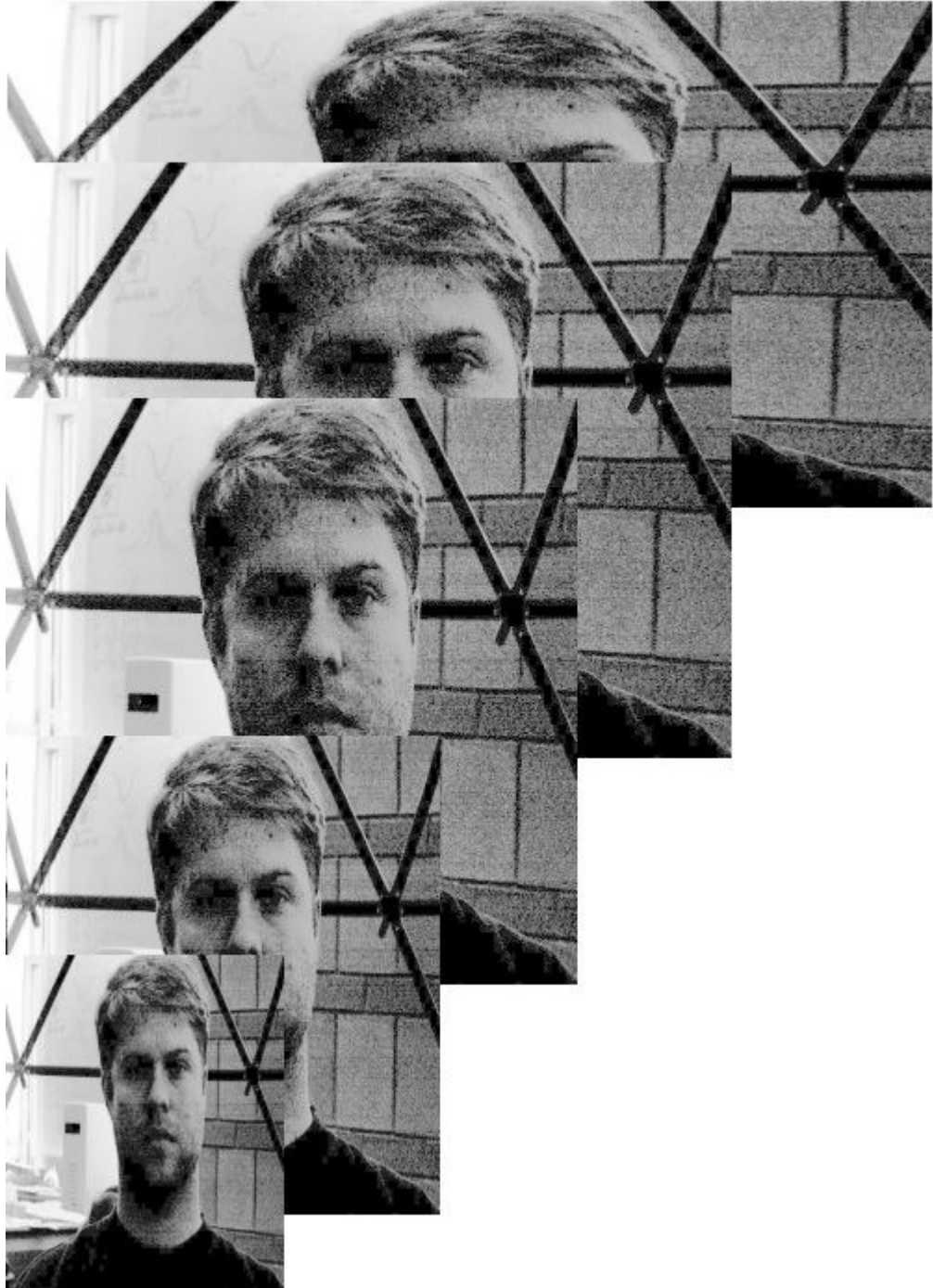


Figure 4.6: Image pyramid of the histogram equalized image.

The last operation for the running part of the neural network during the preprocess operation is applying a 10×10 pixel size window to all locations of the given image. All these subwindows are extracted from the given image. Then mask is applied and pixel values are generated which can be used by the neural network. This is known as image scanning.

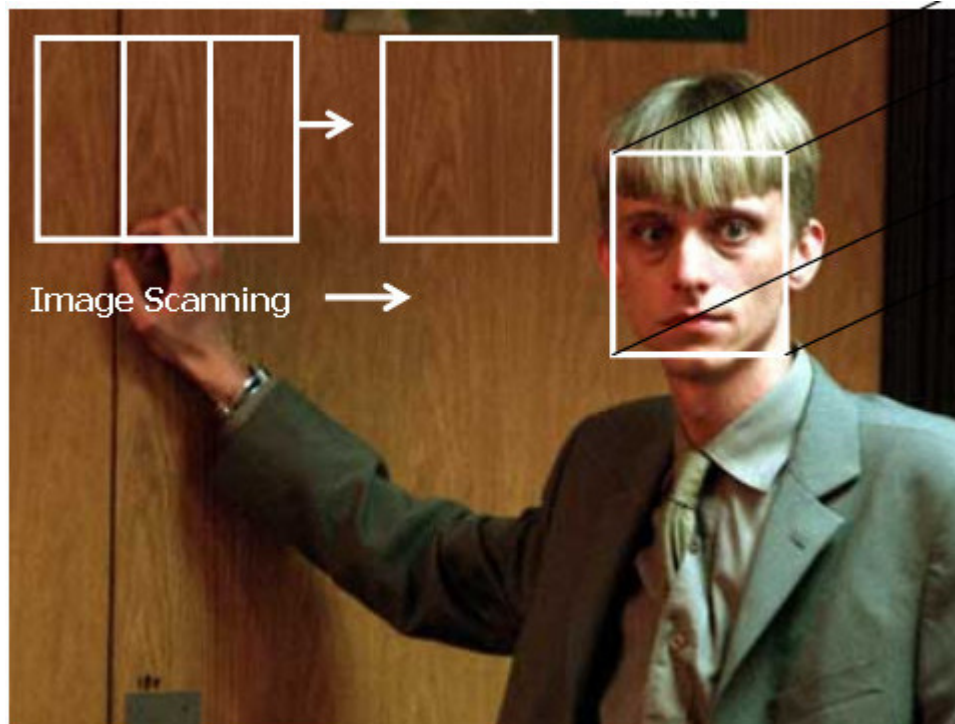


Figure 4.7: Image Scanning

4.2.2 Neural Network Stage

The second component of our system is applying a neural network based filter that receives as input of 10x10 pixel regions of the given image, and generates an output ranging from 1 for faces and 0 for nonfaces. In order to detect faces anywhere in the input image, image pyramid is created and image scanning method is used as I mentioned in the previous section.

After the preprocessing step we have a 10x10 pixel size of image. We passed this window to neural network. Since our neural network works on the pixel values, we have to translate given image to pixel values which will be used by a neural network. This translation is done by the “Neural Network Based Face Detector System” which will be explained in the next section.

4.2.2.1 Neural Network Structure Used for Face Detection.

We used 3 layers feed forward neural network for this system. Neural network is implemented with the consideration of our needs. For example before training and

running we add a translator system which converts given image to txt file. This txt file consists of pixel values of the given image.

The network has 100(10×10) input nodes, and varying size hidden units (some tests were implemented, each network has different size of hidden nodes), and 1 output node which indicates whether or not the window contains a face. The network topology is shown in the Figure 4.8.

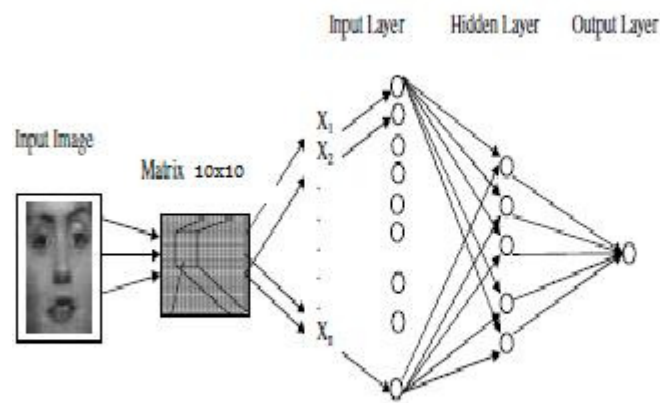


Figure 4.8: Structure of the network

Some faces are detected multiple times as shown in Figure 4.9. Also there are some false detections. Both multiple detections and false detection can be eliminated by the method which will be described in the third step of our system.



Figure 4.9: Multiple detections

4.2.2.2 Training the Neural Network

Training a neural network for face detection is really a challenging because of the variety of face and nonface classes. Large numbers of face and nonface images are needed for training neural network for face detection.

Nearly 7000 face examples are used for training the neural network. These face samples are collected from the face databases such as CMU, Yale etc. and World Wide Web.

These images contained faces of various sizes, orientations, positions, and intensities. A few example images are shown in Figure 4.10. Faces should be extracted from these images in order to use them for training. This extraction operation is done by our system which will be explained in the next section.

Since images of nonface samples is much larger than the space of face images, preparing representative nonface samples for neural network is much more difficult than the preparing the representative face samples.

Instead of collecting the images before training is started, the images are collected during training, in the following manner, adapted from [19]:

1. Create an initial set of nonface images by generating random images. Apply the preprocessing steps to each of these images.

2. Train a neural network to produce an output of 1 for the face examples, and 0 for the nonface examples. The training algorithm is standard error backpropagation. On the first iteration of this loop, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.
3. Run the system on an image of scenery *which contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation > 0.5).
4. Select subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

At the end I have 20000 nonface images. Entirely new network was trained with the examples on which the previous networks had made mistakes

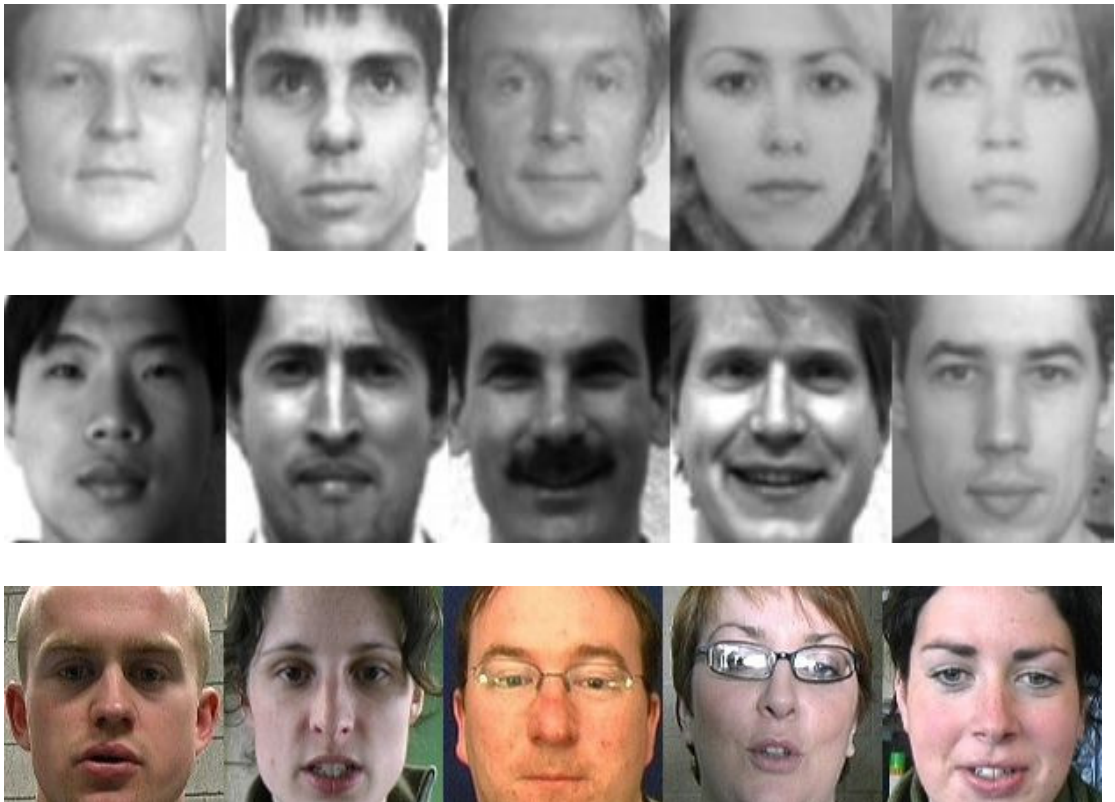


Figure 4.10: Face samples used during training

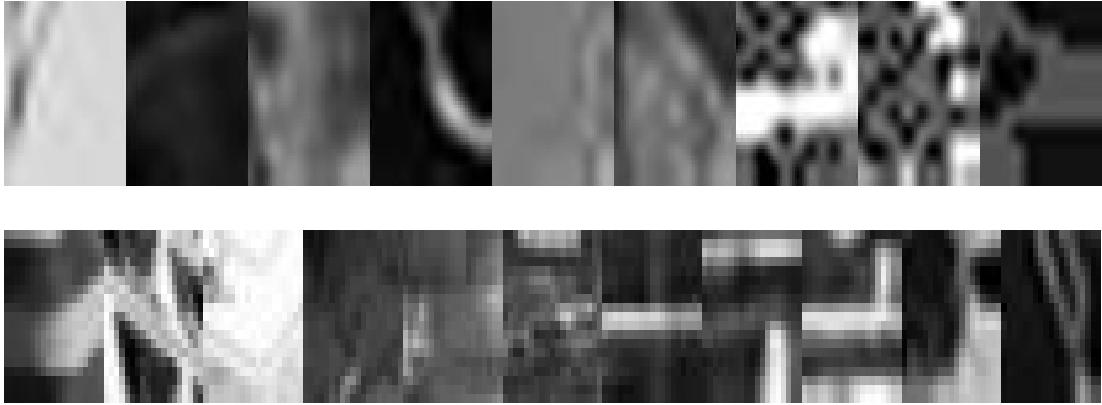


Figure 4.11: Nonface samples used during training

4.2.2 Postprocess

The last stage of our system is postprocessing unit. The raw output from a single network will contain a number of false detections and a face can be detected more than one times as shown in the Fig. 4.12. At the postprocessing step I used two methods in order to reduce the number of false detections and merging overlapping detections

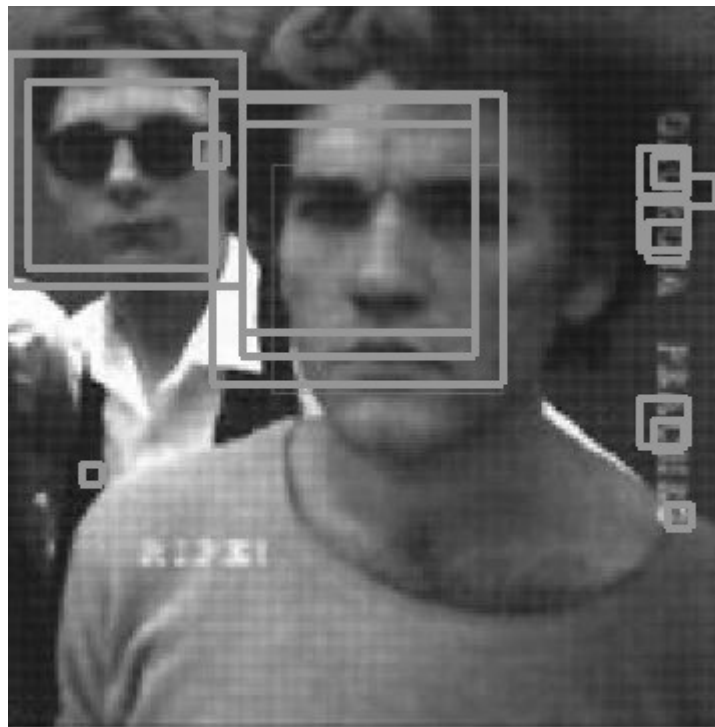


Figure 4.12: Multiple detections with false detections.

4.2.2.1 Merging Overlapping Detections

Since faces are detected at multiple nearby positions or scales and false detections often occur with less consistency as seen in Figure 4.12, false detections can be eliminated by creating a heuristic. The heuristic is that for each location the number of detections within a specified neighborhood of that location can be counted. If the number is above a threshold, then that location is classified as a face. By using this heuristic multiple face detections can be merged and number of false detections decreases.

If a particular location is correctly identified as a face, then all other detection locations which overlap it are likely to be errors, and can therefore be eliminated. Based on the above heuristic regarding nearby detections, we preserve the location with the higher number of detections within a small neighborhood, and eliminate locations with less detection.

The implementation of this heuristic is that explained above in more details.

- Each detection at a particular location and scale is marked in an image pyramid, labeled the “output” pyramid.
- Then, each location in the pyramid is replaced by the number of detections in a specified neighborhood of that location.
- A threshold is applied to these values, and the centroids of all above threshold regions are computed. All detections contributing to a centroid are collapsed down to a single point.
- Each centroid is then examined in order, starting from the ones which had the highest number of detections within the specified neighborhood. If any other centroid locations represent a face overlapping with the current centroid, they are removed from the output pyramid.
- All remaining centroid locations constitute the final detection result.

4.2.2.2 Arbitration among Multiple Networks

In order to reduce number of false positives, and increase the number face detection rate we can apply multiple networks, and arbitrate between their outputs to produce the final decision. Each network is trained in a similar manner, but with random initial weights, as will be seen in the “Experiment” section, the detection and false positive rates of the individual networks will be quite close. However, because of different training conditions and because of self-selection of negative training examples, the networks will make different errors.

The implementation of this method is explained below.

- Firstly an empty table is created with the size of the input image.
- All the Networks run on the input image.
- All Detections are signed in the empty table.
- After running the network, we have a table which consists of all detection locations.
- Final operation is running a heuristic which is explained in the previous section in order to detect false positives and increases the number of detections.

This method not only increases the detection rate but also decreases the number of false detections. Results can be seen in the “Experiment” section.

4.2.2.3 Differences between our System and Previously Proposed System

There are lots of different neural based face detection methods which were previously proposed. They have some differences at preprocessing, neural network and postprocessing steps. Most of these applications differ from each other based on their proposed neural network structure. Some of them used multi hidden layers, some of them use only one hidden layer. Also their network size differs from each other. Moreover proposed algorithms had different number of input nodes varying from 225 to 900. Since proposed networks have huge sizes, they cannot be used for real time applications.

Main difference between our proposed system and other systems is the number of input nodes. The smallest network has 15×15 input nodes. In order to use this neural network for real time applications, the size and the complexity of the network should be decreased. To do this firstly, we created a network which has 3 layers (one input which has 10×10 input nodes, one hidden layer which has 40 hidden layers and one output layer which has just one output node). Since our network has small size running time of our algorithm is decreased significantly.

Another main difference between our system and the previously proposed systems is the number of hidden layers. Most of the previously proposed systems have more than three layers. Proposed systems manually designed the first hidden layer for detecting facial features which may be important for face detection. Since neural networks are really powerful tools, they automatically detect these features and use them for face detection. We thought that there is no need to design a hidden layer just for facial features; it must be done automatically by neural networks just using their capability of solving nonlinear problems.

As a result, our system is the simplest system among all neural network based face detection systems. Using this simplicity our system gained speed however it lost some of face detection capability. More samples will be given in the “Experiment Section”.

Chapter 5

Neural Network Based Face Detection and Image Processing System

5.1 Introduction to Neural Network Based Face Detection and Image Processing System

Neural Network Based Face Detector and Image Processing System which is a command based system, is developed by me in order to use neural network and image processing operations. It is developed in C++ programming language and has more than 5000 lines of code. System is object oriented. In other words, different parts(image processing, file processing, neural network etc) come together in order to create a “Neural Network Based Face Detection and Image Processing System”. This system mainly consists of two parts neural network and image processing. Figure 5.1 is the main screen of the program.

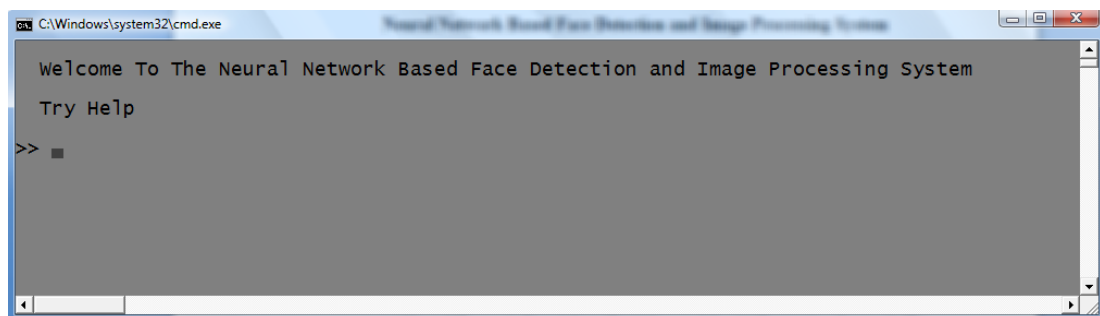


Figure 5.1: Command based, “Neural Network Based Face Detection and Image Processing System”.

5.1.1 Neural Network Part

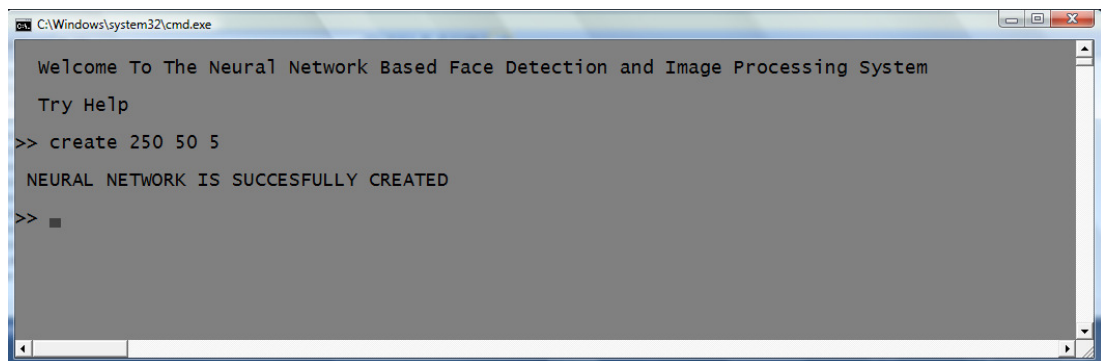
The first part of the system is designed for neural network implementation. In this module a feedforward neural network can be created, trained and used. During the training backpropagation algorithm is used. All the codes in this section were written by me.

Following commands are related to neural network can be used in the system.

- **CREATE:** By using this command user can create a neural network with any size. The syntax of the command is shown below

- o create <# of input nodes> <# of hidden nodes> <# of output nodes>

Figure 5.2 is the output of the create command.



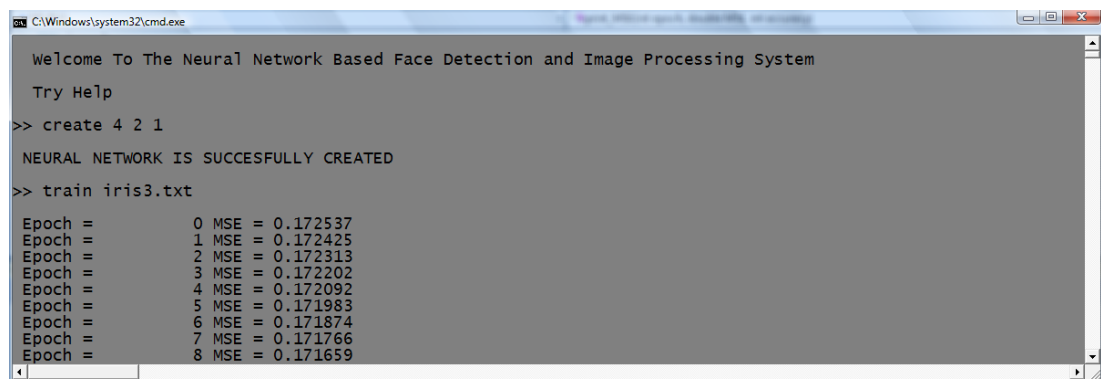
```
C:\Windows\system32\cmd.exe
Welcome To The Neural Network Based Face Detection and Image Processing System
Try Help
>> create 250 50 5
NEURAL NETWORK IS SUCCESSFULLY CREATED
>> █
```

Figure 5.2: Using of create command

- **TRAIN:** After creating a network with random weights, this created network can be trained using the train command. The syntax of the train command is shown below.

- o train <name of the train file>

Figure 5.3 is the output of the train command.

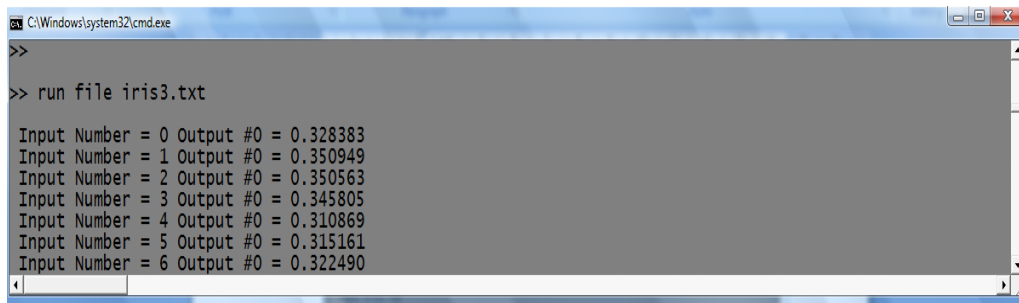


```
C:\Windows\system32\cmd.exe
Welcome To The Neural Network Based Face Detection and Image Processing System
Try Help
>> create 4 2 1
NEURAL NETWORK IS SUCCESSFULLY CREATED
>> train iris3.txt
Epoch =      0 MSE = 0.172537
Epoch =      1 MSE = 0.172425
Epoch =      2 MSE = 0.172313
Epoch =      3 MSE = 0.172202
Epoch =      4 MSE = 0.172092
Epoch =      5 MSE = 0.171983
Epoch =      6 MSE = 0.171874
Epoch =      7 MSE = 0.171766
Epoch =      8 MSE = 0.171659
```

Figure 5.3: Using of train command

- **RUN:** After training the network we want to run this network with given input file or image. RUN command contains subcommands. Run command can be used with a image, images under a specific folder or a file.
 - In order to use neural network on a file the following syntax should be used.
 - run file <name of the file>

Figure 5.4 is the output of the run from a specific file.

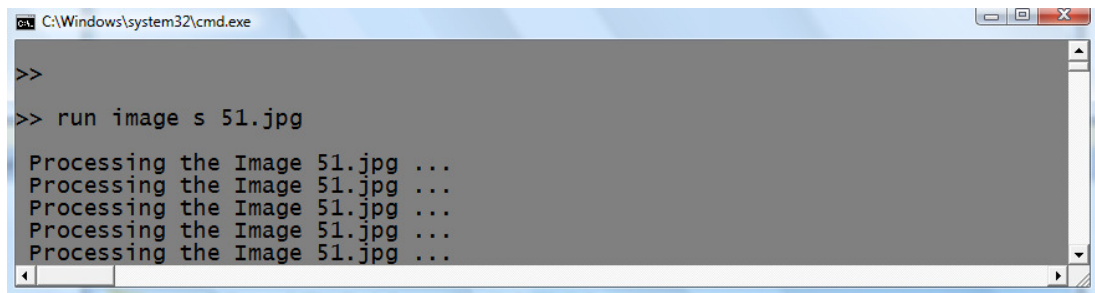


```
C:\Windows\system32\cmd.exe
>>
>> run file iris3.txt
Input Number = 0 Output #0 = 0.328383
Input Number = 1 Output #0 = 0.350949
Input Number = 2 Output #0 = 0.350563
Input Number = 3 Output #0 = 0.345805
Input Number = 4 Output #0 = 0.310869
Input Number = 5 Output #0 = 0.315161
Input Number = 6 Output #0 = 0.322490
```

Figure 5.4: Using the run command with a file

- In order to use neural network with a single image following syntax should be used
 - run image s <name of the image>

Figure 5.5 is the output of the run command from an image.

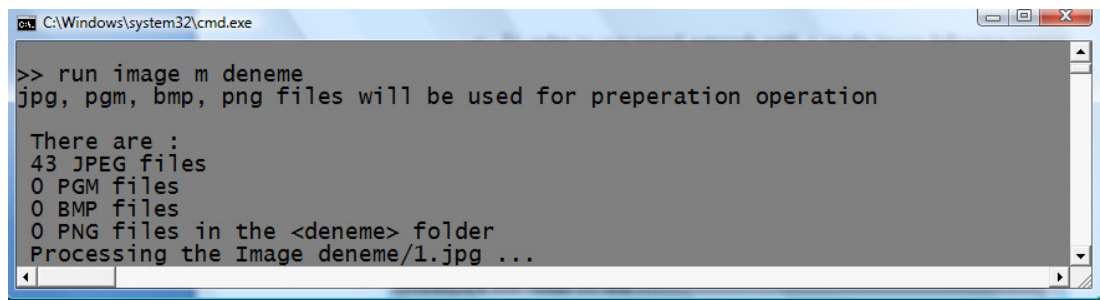


```
C:\Windows\system32\cmd.exe
>>
>> run image s 51.jpg
Processing the Image 51.jpg ...
Processing the Image 51.jpg ...
Processing the Image 51.jpg ...
Processing the Image 51.jpg ...
Processing the Image 51.jpg ...
```

Figure 5.5: Using run command with single image

- In order to use neural network with images under a specific folder following command syntax should be used.
 - run image m <name of the folder>

Figure 5.6 is the output of the run command from a specific directory.



```
C:\Windows\system32\cmd.exe
>> run image m deneme
jpg, pgm, bmp, png files will be used for preperation operation

There are :
43 JPEG files
0 PGM files
0 BMP files
0 PNG files in the <deneme> folder
Processing the Image deneme/1.jpg ...
```

Figure 5.6: Using run command with images under a specific folder

- **SAVE:** The network which is trained can be used later by using the save command. The syntax of the save command is shown below.

- save <desired name for the network>

Figure 5.7 is the output of the save command.



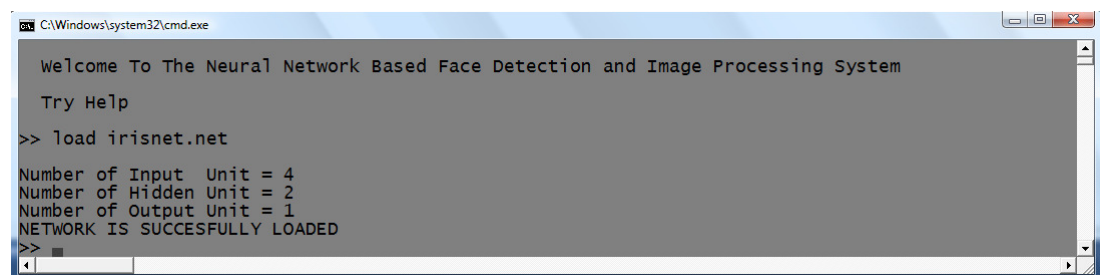
```
C:\Windows\system32\cmd.exe
>> save irisnet.net
NETWORK SUCCESFULLY SAVED
>>
```

Figure 5.7: Using the save command.

- **LOAD:** The previously saved network can be used by using the load command. The syntax of this command is shown below.

- load <name of the previously saved network>

Figure 5.8 is the output of the load command.



```
C:\Windows\system32\cmd.exe
Welcome To The Neural Network Based Face Detection and Image Processing System
Try Help
>> load irisnet.net
Number of Input Unit = 4
Number of Hidden Unit = 2
Number of Output Unit = 1
NETWORK IS SUCCESFULLY LOADED
>>
```

Figure 5.8: Using the load command.

- Also there are some commands which are using for setting and getting the values of variables used by the network. They are:

- epoch
- learning rate

- accuracy
- threshold

5.1.2 Image Processing Part

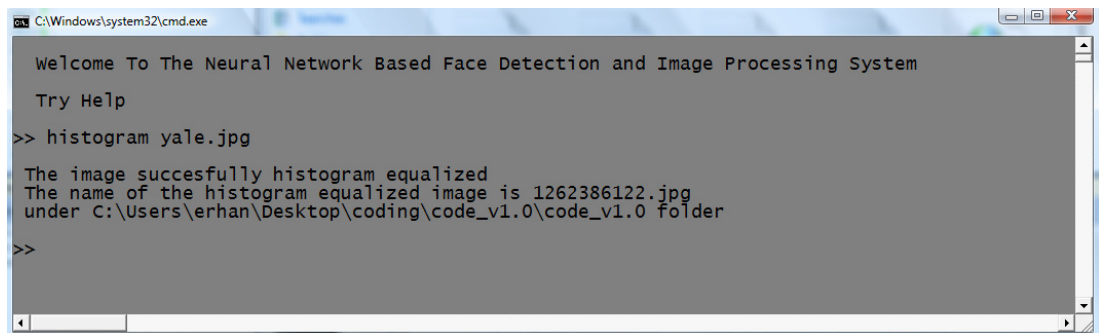
The second part of the Neural Network Based Face Detection and Image Processing System is image processing part. In this part there are lots of image processing algorithms such as histogram equalization; morphological gradient, thresholding etc. are implemented.

Some of the image processing related commands are shown below.

- **HISTOGRAM:** By using this command user can apply histogram equalization algorithm to the given image. The syntax of the histogram command is shown below.

- histogram <name of the image>

Figures 5.9 and 5.10 are the output of the histogram command



```
C:\Windows\system32\cmd.exe

Welcome To The Neural Network Based Face Detection and Image Processing System
Try Help
>> histogram yale.jpg

The image succesfully histogram equalized
The name of the histogram equalized image is 1262386122.jpg
under C:\Users\erhan\Desktop\coding\code_v1.0\code_v1.0 folder

>>
```

Figure 5.9: Using the histogram command.



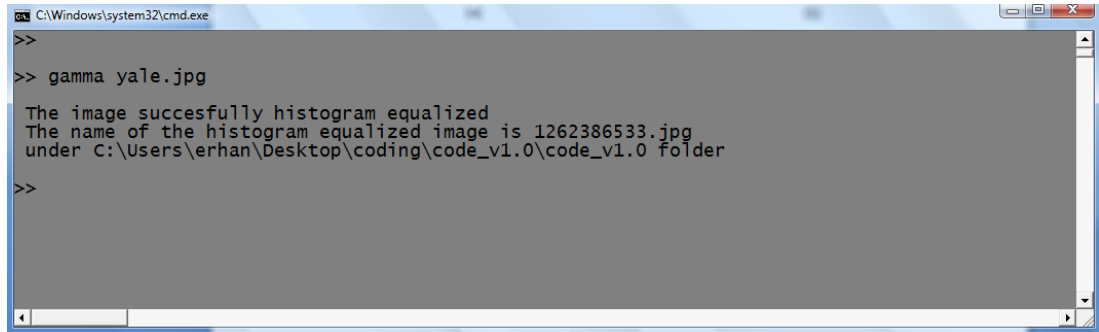
(a)

(b)

Figure 5.10: Result of the histogram command. (a) original, (b) histogram equalized image

- **GAMMA:** Gamma correction on a given image can be applied by using the gamma command. The syntax of this command is shown below.
 - o gamma <name of the input image>

Figures 5.11 and 5.12 are output of the gamma commands.



```
C:\Windows\system32\cmd.exe
>>
>> gamma yale.jpg
The image succesfully histogram equalized
The name of the histogram equalized image is 1262386533.jpg
under C:\Users\erhan\Desktop\coding\code_v1.0\code_v1.0 folder
>>
```

Figure 5.11: Using the gamma command.



(a)

(b)

Figure 5.12: Result of the gamma command. (a) original image, (b) gamma corrected image

- **GRADIENT:** Morphological gradient operation is applied to the given image via gradient command. The syntax of this given image is shown below.
 - o Gradient <name of the image>

Figures 5.13 and 5.14 are output of the gradient command.

```
C:\Windows\system32\cmd.exe
Welcome To The Neural Network Based Face Detection and Image Processing System
Try Help
>> gradient 12.jpg
GRADIENT OPERATION IS SUCCESFULLY APPLIED TO GIVEN IMAGE
THE NAME OF THE RESULT IMAGE IS 1262386883.jpg
>> █
```

Figure 5.13: Usage of the gradient command.



(a)

(b)

Figure 5.14: Result of the gradient command. (a) original image, (b) result image

Also there are another commands such as preprocess, debug, print, show, rename, dir, cd etc which are not described here. There are 34 commands which are defined in this system.

Chapter 6

Experimental Results

A number of experiments were performed to test the system. The system was tested on two large sets of images. Test Set 1 consists of 39 images, including images from the World Wide Web, scanned from photographs and newspaper pictures, and digitized from broadcast television. The images contain a total of 203 frontal faces, and require the networks to examine 10,511,877 10×10 pixel windows. Test Set 2 consists of 25 images from the different face databases. The images contain 190 frontal faces, and require the networks to examine 6,106,484 10×10 pixel windows.

The outputs from our face detection networks are not binary. The neural network produces real values between 1 and 0, indicating whether or not the input contains a face. A threshold value of 0.5 is used during *training* to select the negative examples (if the network outputs a value of greater than zero for any input from a scenery image, it is considered a mistake). Although this value is intuitively reasonable, by changing this value during testing, we can vary how conservative the system is. To examine the effect of this threshold value during testing, we measured the detection and false positive rates as the threshold was varied from 0.5 to 1. At a threshold of 1, the false detection rate is zero, but no faces are detected. As the threshold is decreased, the number of correct detections will increase, but so will the number of false detections. This tradeoff is presented in the following tables and charts.

Also we used three different networks during the experiments. These networks have different number of hidden layers and they have different accuracies. Also their training sets are almost different from each other. Table 6.1 shows these three networks.

Name of the Network	# of Hidden Nodes	Accuracy
Network 1	39	0,001142
Network 2	40	0,001391
Network 3	39	0,001548

Table 6.1: Network used during the experiment section

In the next section I will give the experimental results of these three networks on two different datasets.

6.1 Face Threshold

Face threshold is using in order to indicate whether the output of the neural network is a face or nonface. I have conducted several experiments on two datasets with 3 different networks while the face threshold value varied from 0.5 to 1.0. Below tables and charts represent the experimental results.

- Network 1 on Dataset 1

NETWORK 1		
Threshold	Face Detection Rate	False Detection Rate
0.5	49	0,00006336
0.6	37	0,00003586
0.7	28	0,00002283
0.8	19	0,00001113
0.9	9	0,00000171
1.0	0	0

Table 6.2: Face Detection and False Detection Rates as the threshold varied

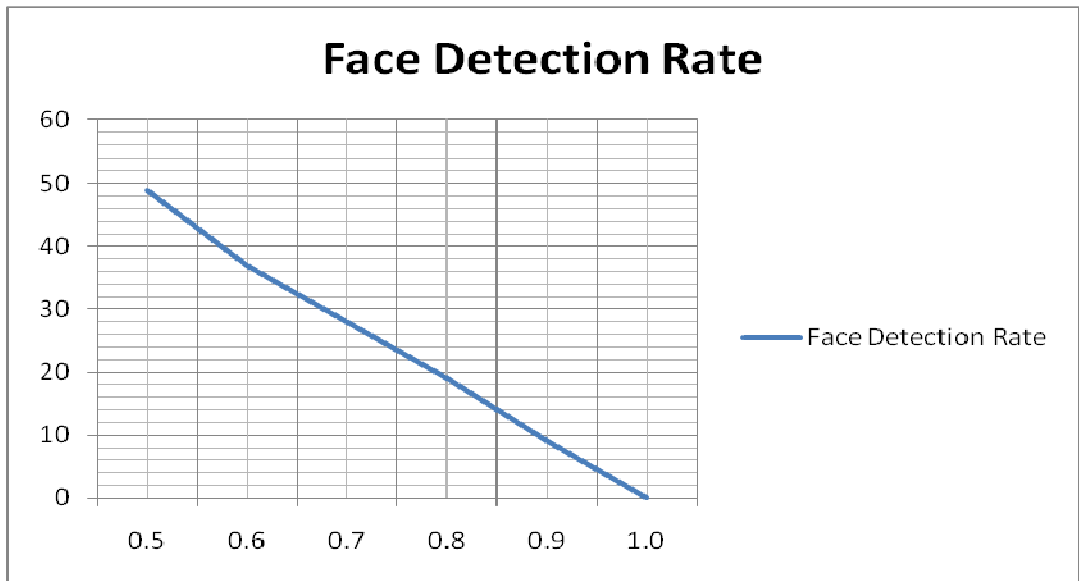


Figure 6.1: Face Detection Rate against the threshold value.

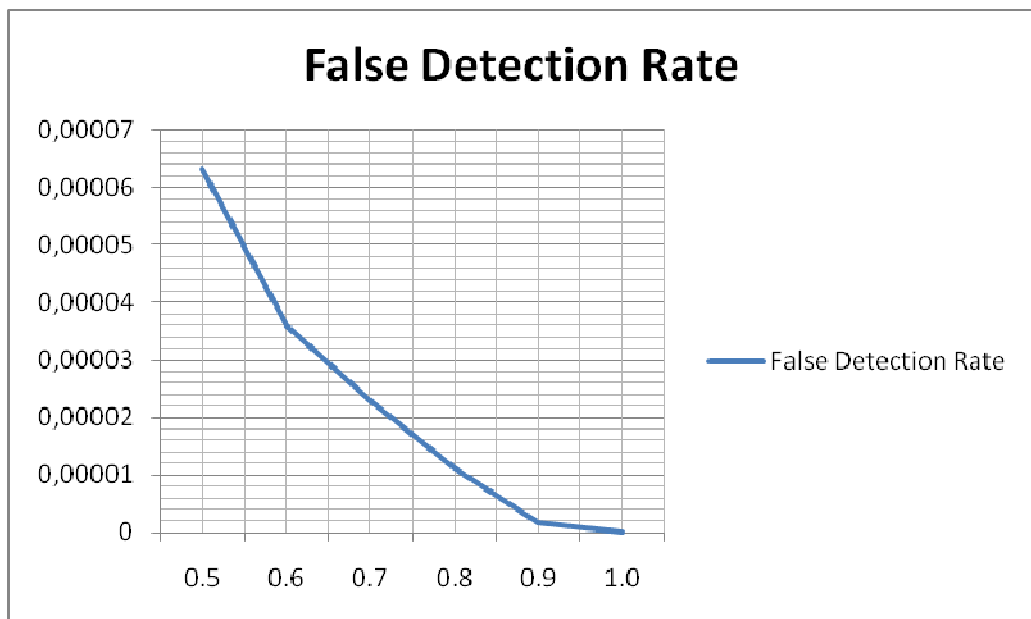


Figure 6.2: False Detection Rate against threshold value

- Network 1 on Dataset 2

NETWORK 1		
Threshold	Face Detection Rate	False Detection Rate
0.5	50	0,007287336
0.6	36,8	0,003406215
0.7	30,5	0,002096133
0.8	21,6	0,00103169
0.9	11,6	0,00004912811
1.0	0	0

Table 6.3: Face Detection and False Detection Rates as the threshold varied

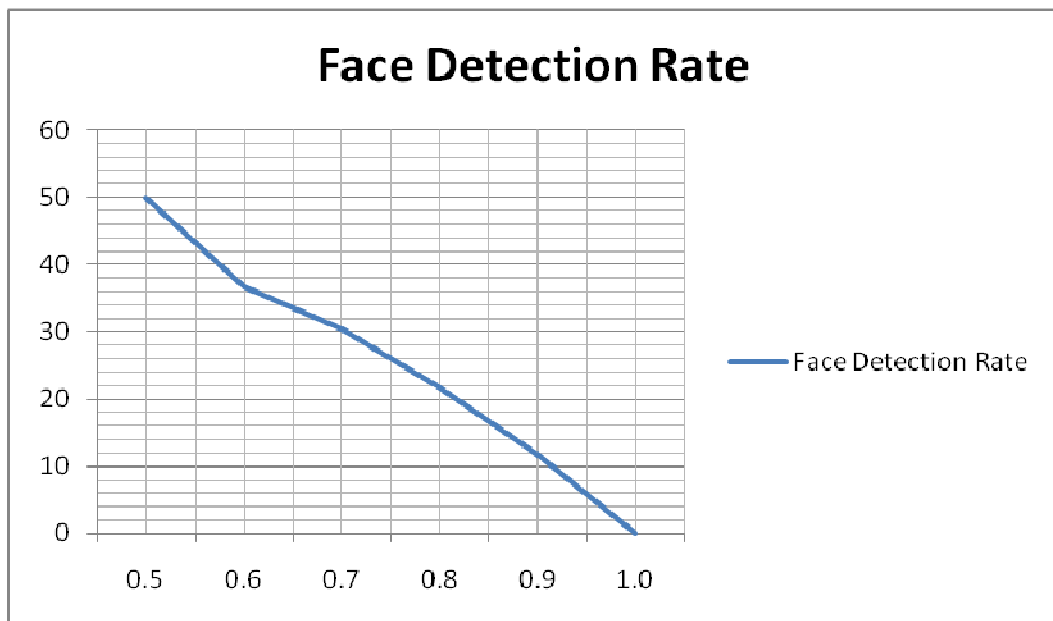


Figure 6.3: Face Detection Rate against the threshold value.

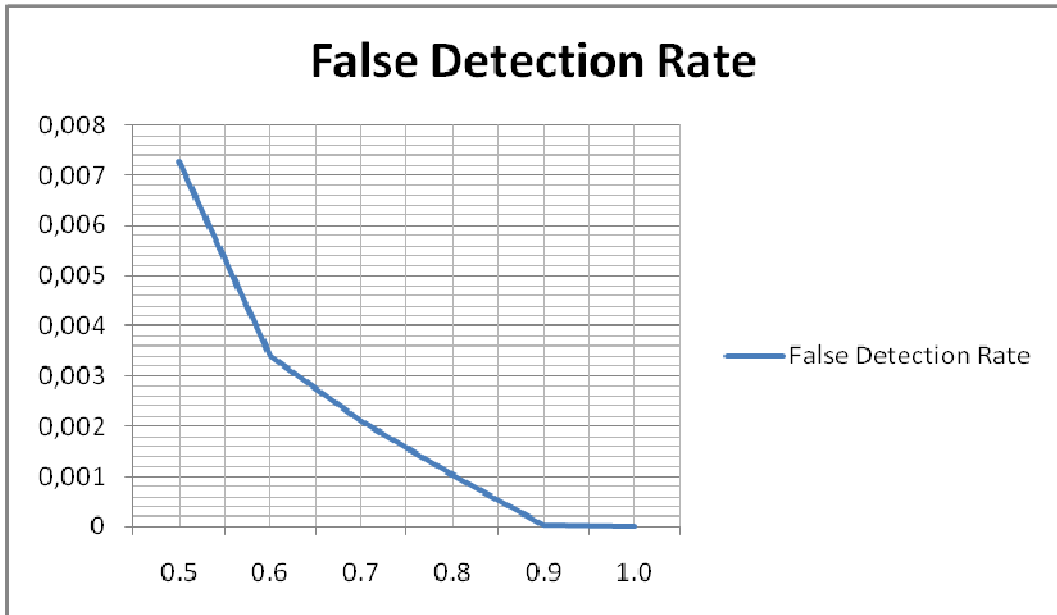


Figure 6.4: False Detection Rate against threshold value

- **Network 2 on Dataset 1**

NETWORK 2		
Threshold	Face Detection Rate	False Detection Rate
0.5	50	0,00003548
0.6	44	0,00002483
0.7	39	0,00002102
0.8	35	0,00001313
0.9	27	0,00000504000
1.0	0	0

Table 6.4: Face Detection and False Detection Rates as the threshold varied

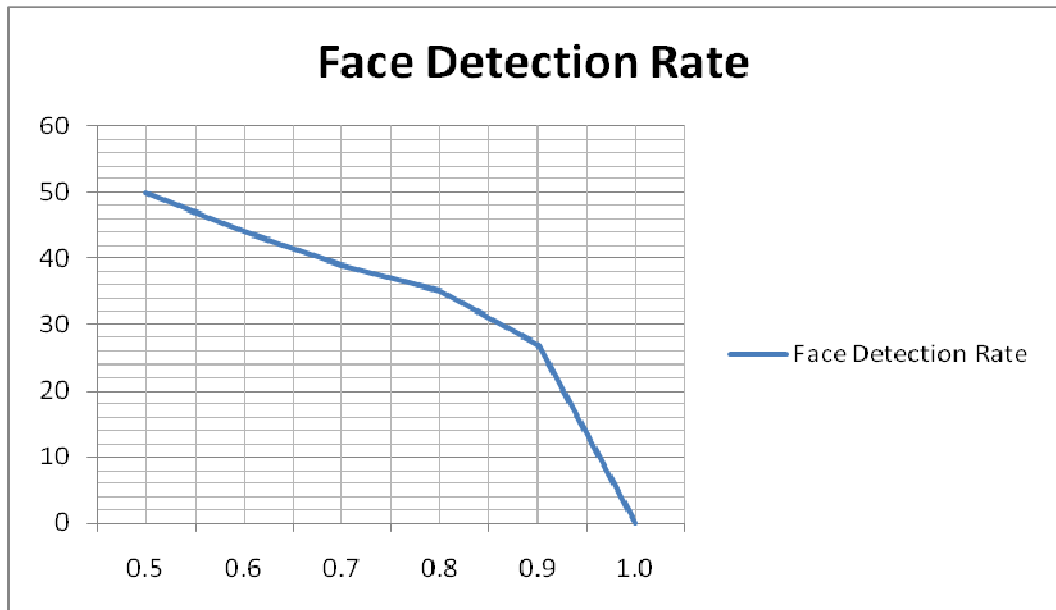


Figure 6.5: Face Detection Rate against the threshold value.

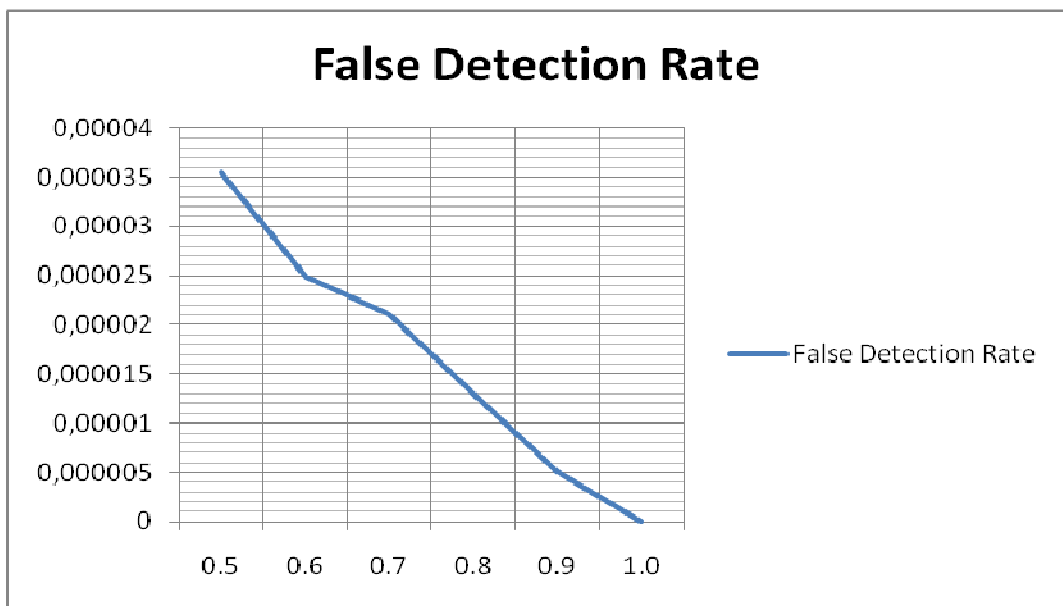


Figure 6.6: False Detection Rate against threshold value

- Network 2 on Dataset 2

NETWORK 2		
Threshold	Face Detection Rate	False Detection Rate
0.5	50.8	0,007287336
0.6	45	0,001932372
0.7	44	0,001129946
0.8	33.5	0,000605913
0.9	22,5	0,00011463225
1.0	0	0

Table 6.5: Face Detection and False Detection Rates as the threshold varied

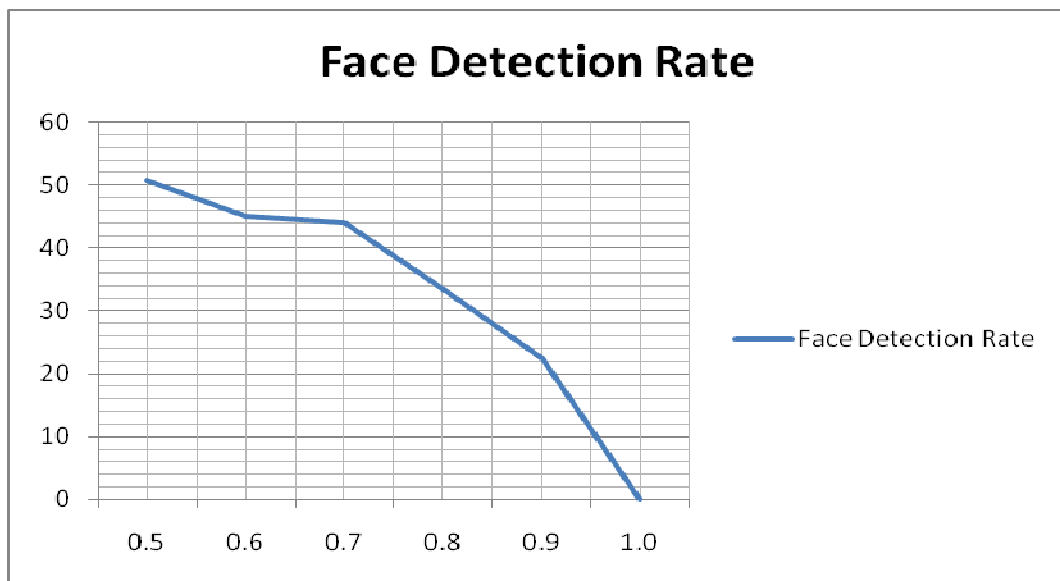


Figure 6.7: Face Detection Rate against the threshold value.

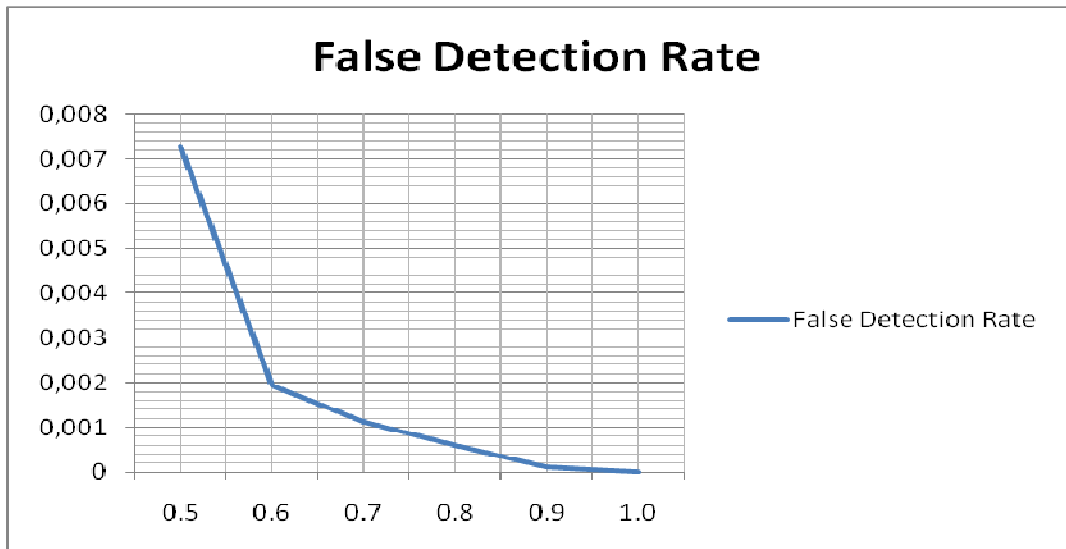


Figure 6.8: False Detection Rate against threshold value

6.2 Multiple Networks

I have implemented OR operation on the result of more than one neural network in order to increase face detection rate. Experiments show that face detection rate increased after the arbitrating among multiple networks. However, number of false detection rate also increased.

- 2 Networks on Dataset 1

2 NETWORKS		
Threshold	Face Detection Rate	# of False Detections
0.5	63	0,00010426
0.6	46	0,00004186
0.7	34	0,00002435
0.8	22	0,00001161
0.9	8	0,00000143
1.0	0	0

Table 6.6: Face Detection and False Detection Rates as the threshold varied

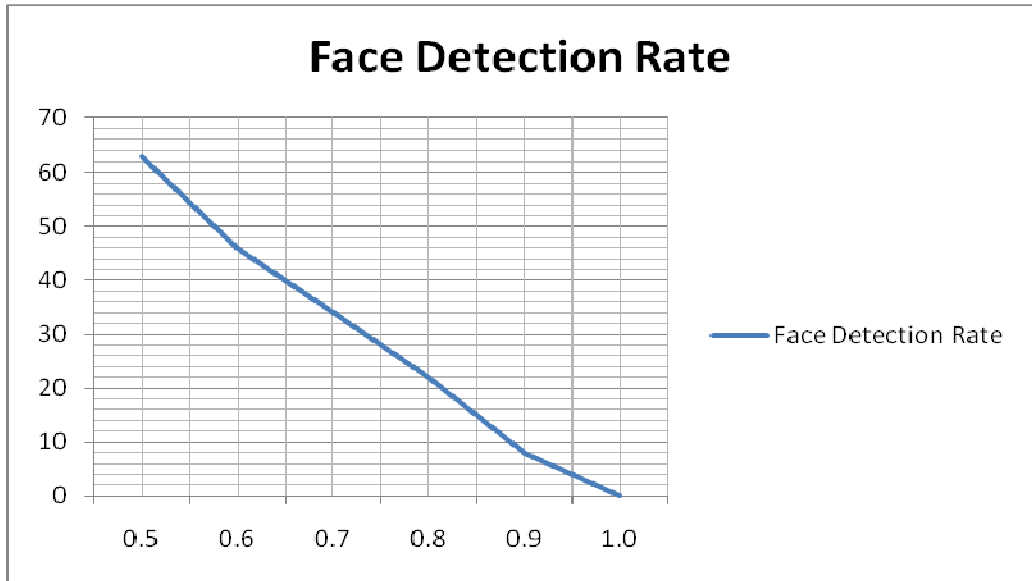


Figure 6.9: Face Detection Rate against the threshold value

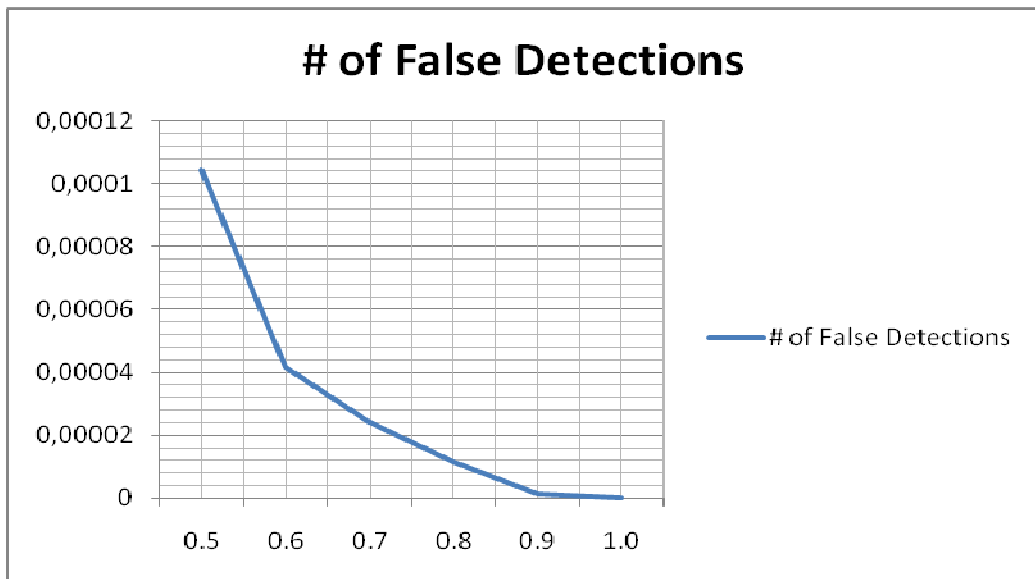


Figure 6.10: False Detection Rate against the threshold value

- 2 Networks on Dataset 2

2 NETWORKS		
Threshold	Face Detection Rate	# of False Detections
0.5	61,6	0,006026381
0.6	45,9	0,002882182
0.7	37,9	0,001686732
0.8	25,80	0,000736922
0.9	12,80	0,0000327521
1.0	0,00	0

Table 6.7: Face Detection and False Detection Rates as the threshold varied

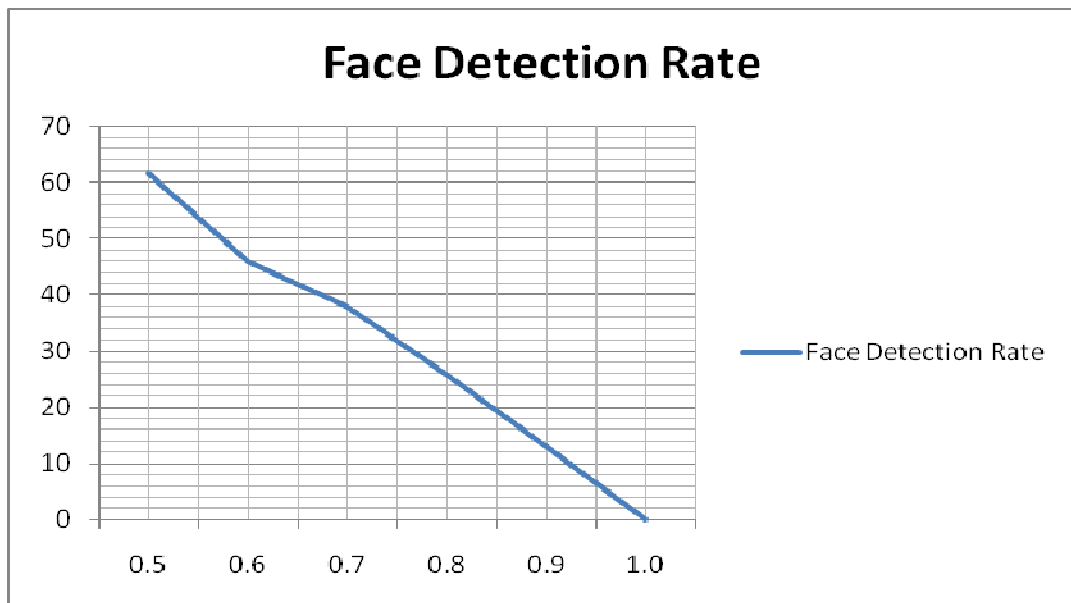


Figure 6.11: Face Detection Rate against the threshold value

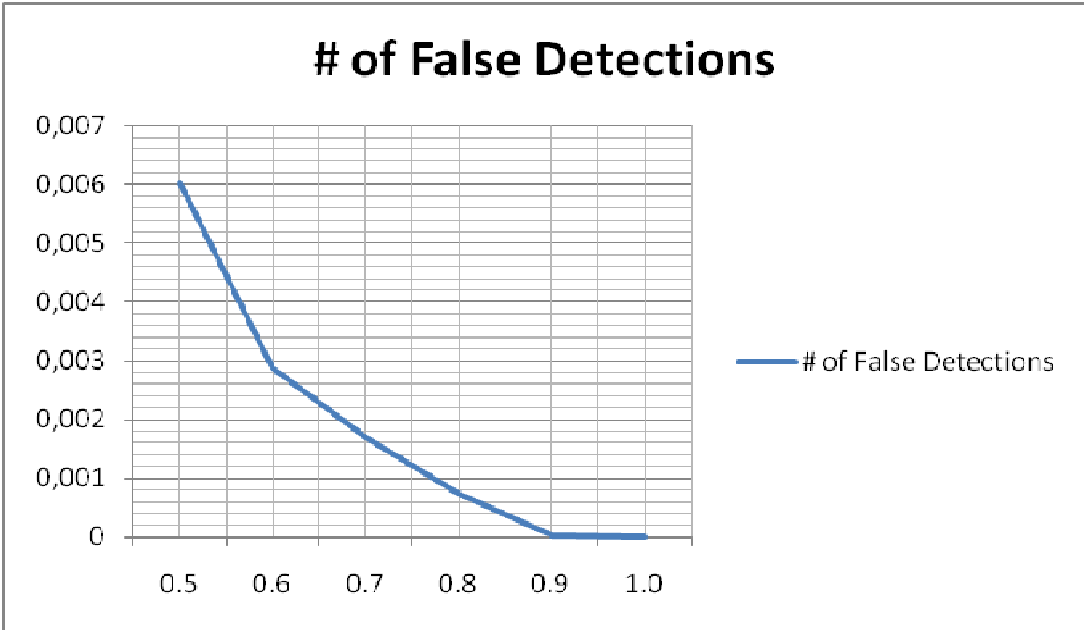


Figure 6.12: False Detection Rate against the threshold value



Figure 6.13: Obtained outputs

Conclusion

In this thesis neural network based face detection method which was the simplest neural network based method in the literature was proposed. I made lots of experiments related to variables during the running and training of the neural network. These variables have different affects on results. The most affective variable on the result of the detection rate is threshold value which is used for deciding whether a output of a neural network is a face or nonface. Experiments show that if the threshold value set to high value, the false detection rate decreases. At the same time face detection rate also decreases. If the threshold value is set to low value, number of false detections increases. Also the face detection rate increases.

During the testing part of the algorithm I used more than 3 networks. Their results were different from each other. Also I applied OR operation to the results of two and three neural networks. Result of this operation shows that, both face detection rate number of false detection rates increases.

All in all, result of this algorithm is as good as the previously proposed algorithms. One of the reasons of this failure is that the capability of the simple system is not enough to detect complex faces. Also further training will increase the face detection rate and decreases the false detection rate. Moreover more heuristics will decrease the number of false detections.

References

- [1] Yang M., Kriegman D., Ahuja N., “Detecting Faces in Images : A Survey”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, 2002.
- [2] Yang M.-H., Roth D., and Ahuja N., “A SNoW-Based Face Detector,” *Advances in Neural Information Processing Systems 12*, S.A. Solla, T. K. Leen, and K.-R. Muller, eds., pp. 855-861, MIT Press, 2000.
- [3] Sirohey S.A., *Human Face Segmentation and Identification*, Technical Report CS-TR-3176, Univ. of Maryland, 1993.
- [4] Graf H.P., Chen T., Petajan E., and Cosatto E., “Locating Faces and Facial Parts,” *Proc. First Int’l Workshop Automatic Face and Gesture Recognition*, pp. 41-46, 1995.
- [5] Han C.-C., Liao H.-Y.M., Yu K.-C., and Chen L.-H., “Fast Face Detection via Morphology-Based Pre-Processing,” *Proc. Ninth Int’l Conf. Image Analysis and Processing*, pp. 469-476, 1998.
- [6] Augusteijn M.F. and Skujca T.L., “Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology,” *Proc. IEEE Conf. Neural Networks*, pp. 392-398, 1993.
- [7] Seow M. J., Valaparla D., and Asari V. K., “Neural Network Based Skin Color Model for Face Detection,” in *Applied Imagery Pattern Recognition Workshop*, 2003, pp. 141–145.
- [8] Sobottka K. and Pitas I., “Face Localization and Feature Extraction Based on Shape and Color Information,” *Proc. IEEE Int’l Conf. Image Processing*, pp. 483-486, 1996.
- [9] Sakai T., Nagao M., and Fujibayashi S., “Line Extraction and Pattern Detection in a Photograph,” *Pattern Recognition*, vol. 1, pp. 233-248, 1969.
- [10] Govindaraju V., Sher D.B., Srihari R.K., and Srihari S.N., “Locating Human Faces in Newspaper Photographs,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 549-554, 1989.

- [11] Yuille A., Hallinan P., and Cohen D., "Feature Extraction from Faces Using Deformable Templates," *Int'l J. Computer Vision*, vol. 8, no. 2, pp. 99-111, 1992.
- [12] Lanitis A., Taylor C.J., and Cootes T.F., "An Automatic Face Identification System Using Flexible Appearance Models," *Image and Vision Computing*, vol. 13, no. 5, pp. 393-401, 1995.
- [13] Kirby M. and Sirovich L., "Application of the Karhunen-Loe`ve Procedure for the Characterization of Human Faces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, Jan. 1990.
- [14] Kohonen T., *Self-Organization and Associative Memory*, Springer 1989.
- [15] Turk M. and Pentland A., "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.
- [16] Sung K.-K. and Poggio T., *Example-Based Learning for View- Based Human Face Detection*, Technical Report AI Memo 1521, Massachusetts Inst. of Technology AI Lab, 1994.
- [17] Agui T., Kokubo Y., Nagashashi H., and Nagao T., "Extraction of Face Recognition from Monochromatic Photographs Using Neural Networks," *Proc. Second Int'l Conf. Automation, Robotics, and Computer Vision*, vol. 1, pp. 18.8.1-18.8.5, 1992.
- [18] Propp M. and Samal A., "Artificial Neural Network Architectures for Human Face Detection," *Intelligent Eng. Systems through Artificial Neural Networks*, vol. 2, 1992.
- [19] Soulie F., Viennet E., and Lamy B., "Multi-Modular Neural Network Architectures: Pattern Recognition Applications in Optical Character Recognition and Human Face Recognition," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 721-755, 1993.
- [20] Rowley H., Baluja S., and Kanade T., "Neural Network-Based Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 203-208, 1996.
- [21] Osuna E., Freund R., and Girosi F., "Training Support Vector Machines: An Application to Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130-136, 1997.
- [22] Yang G. and Huang T. S., "Human Face Detection in Complex Background," *Pattern Recognition*, vol. 27, no. 1, pp. 53-63, 1994.
- [23] Schneiderman H. and Kanade T., "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 45-51, 1998.

- [24] Rikert T., Jones M., and Viola P., "A Cluster-Based Statistical Model for Object Detection," *Proc. Seventh IEEE Int'l Conf. Computer Vision*, vol. 2, pp. 1046-1053, 1999.
- [25] Heaton J., *Programming Neural Networks with Java*, 1st Edition, 2005.
- [26] Anifantis D., Dermatas E. and Kokkinakis G., *A Neural Network Method for Accurate Face Detection on Arbitrary Images*, 2005.
- [27] Garcia C. and Delakis M., *A Neural Architecture for Fast and Robust Face Detection*, 2001.
- [28] FeÂ raud R., O. Bernier J., Viallet J., and Collobert M., "A Fast and Accurate Face Detector Based on Neural Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, january 2001.

Curriculum Vitae