DOI: http://dx.doi.org/10.18185/eufbed.95389

APPLICATION OF A MULTI LAYER PERCEPTRON NETWORK FOR DETECTION OF HUMAN FACES ON A LARGE DATABASE

ÇOK KATMANLI BİR YAPAY SİNİR AĞI SİSTEMİNİN RESİMLER İÇERİSİNDE İNSAN YÜZÜ BULMAK AMACIYLA ÇOK SAYIDA İNSAN YÜZÜ İÇEREN BİR VERİ TABANINA UYGULANMASI

Metin SALİHMUHSİN[1*], Yavuz Selim İŞLER[1]

[1] Department of Electrical and Electronic, Faculty of Engineering & Architecture, Kahramanmaras Sutcu Imam University, Kahramanmaras, TURKEY

Geliş Tarihi (Received): 13/10/2014 **Kabul Tarihi (Accepted)**: 23/03/2015

ABSTRACT

In this paper, we have designed and developed a face detection system that could detect human faces of different sizes and orientations in gray scale image files. The developed system consists of a camera, a computer, an image acquisition setup and a face detection method written in Matlab. In order to test detection capability of the system, we have collected a database that contains 125 different images of 125 different people with the above mentioned image acquisition setup. A program based on a multilayer perceptron network (MLP) is developed in order to detect faces in images. The MLP based face detection program works in two levels and uses the symmetry inherited in a human face between left and right sides. In the first level, the detection system searches for a similar structure to a right side of a human faces in the image using a predefined face template in multi resolution. When the detection method finds such a region, it processes that portion of the image further in order to find whether the other side of the face exists or not in the closed vicinity. We have tested our system on a database that contains 100 images selected from above mentioned database. The remaining 25 images are used to form training set for the MLP algorithm. Simulation results show that the method performs %81 correct detection rate on the test set.

Keywords: Face detection system; Face image database; Artificial neural networks

ÖZET

Bu yayında gri seviyeli resimler içerisindeki farklı boyut ve konumlardaki insan yüzlerini bulan bir yüz bulma sisteminin tasarlanıp geliştirilmesi anlatılmaktadır. Geliştirmiş olduğumuz sistem bir kamera, bir bilgisayar, bir resim alma seti ve Matlab pragraminda yazılmış bir yüz bulma algoritmasından oluşmaktadır. Yüz bulma metodunun başarısını test edebilmek amacıyla içerisinde 125 farklı kişinin 125 resminin bulunduğu bir

***Sorumlu Yazar:** *metinsalihmuhsin@yahoo.com*

insan yüzleri veri tabanı geliştirmiş olduğumuz resim alma seti kullanılarak oluşturulmuştur. Resim dosyaları içerisindeki insan yüzü kısımlarını bulmak için çok katmanlı bir yapay sinir ağı programı geliştirilmiştir. Yapay sinir ağları temelli bu yüz bulma programı iki aşamalı olarak yüz bulma işlemini gerçekleştirmektedir ve bu amaçla insan yüzünün sağ ve sol taraflarının doğal olarak birbirlerine benzemesi özelliğini kullanmıştır. İlk aşamada resim içerisinde insan yüzünün sağ tarafına benzeyen kısımlarının bulunması işlemi daha önceden hazırlanan bir insan yüzü taslağı kullanılarak gerçeklenmiş ve bu işlem resime farklı çözünürlüklerde uygulanmıştır. Metod resim içerisinde aranılan özelliklerde bir kısım bulduğunda ikinci aşamaya geçmekte ve ikinci aşamada bulunan kısmın çevresinde yüz kısmının diğer tarafının olup olmadığını bulmaktadır. Geliştirilen yüz tanıma sistemi, bu çalışma için oluşturduğumuz veri tabanından 100 kişinin resimlerinin bulundugu bir test veri tabanı üzerinda test edilmiştir. Geriye kalan 25 kişinin resimleri ise yapay sinir ağını eğitmek üzere kullanılmıştır. Simulasyon sonuçları metodumuzun test seti üzerinde %81 oranında bir doğru tanıma başarısı gösterdiğini ortaya koymuştur.

Anahtar Kelimeler: Yüz tanıma sistemi; İnsan yüzü veri tabanı; Yapay sinir ağları

1. INTRODUCTION

Technological advances in electronics and vision systems brought by itself the need for security/surveillance systems, secure access to machines and financial accounts, identification systems etc. As a result, biometric identification systems gained a lot of attention in recent years. Face detection and identification, finger print recognition and iris recognition are among most popular biometric identification systems. Face detection finds its most use in security/surveillance and automated face recognition systems. In both applications, detecting a face is the first step before taking an action. Many approaches are developed to detect faces in still images and video sequences. Some of these methods were based on feature extraction and parameterization techniques such as Fourier, wavelets and PCA (Direkoglu et al. 2005; Yacoub et al. 1999; Nanni et al. 2007). Some others were based on probabilistic, statistical and image processing based approaches (Lin et al. 1997; Moghaddam et al. 1997; Cootes et al. 1996; Han et al. 1998) and some others were solely neural network based (İşler et al. 2008; Feraud et al. 2001; Rowley et al. 1996). A good survey of these methods can be found in Yang et al

2002. Here we will brief one of the above methods namely Probabilistic Decision Based Neural Networks (PDBNN).

Lin et al 1997 used PDBNN to detect and classify faces in video sequences. Their system was consisted of four main stages: Face detector, eye localizer, feature extractor and face recognizer. The PDBNN is developed based on Decision Based Neural Networks (DBNN). It consists of as many subnets as number of different classes that the network has to differentiate. The learning rule that is used in the PDBNN is a decision based learning rule, i.e. the teacher value in training phase only gives a merit of correctness of a classification. Training of the PDBNN is accomplished mainly in two phases: Locally unsupervised learning phase (LU) and globally supervised learning phase (GS). The LU phase is used to train each subnet to learn its corresponding class. After the LU phase completes, the network performs the GS phase to form boundaries in between each subnet. Here, the network only uses misclassified patterns to fine tune the decision boundaries of subnets.

They have tested their system on a database consisted of 40 people. Each person had 60 images. Images of each person had taken as he/she was slowly moving his/her head. The network detected faces correctly for %75 of the images.

2. MATERIAL AND METHOD

Database used in this work is collected by an image acquisition system we have developed. A Microsoft web-cam is used as a camera. The connection between camera and computer is accomplished by Matlab's GUIDE toolbox. Images were taken from many different backgrounds. All pictures are taken as gray level images of 240x320 pixels.

The database consisted of 125 different images of 125 individuals. Some images contain only one face whereas some others include several faces each located at a different distance from the camera. A few images contain certain degree of tilt and rotation of faces up to 20 to 25 degrees. Figure 1 shows images of 4 individuals from our database.

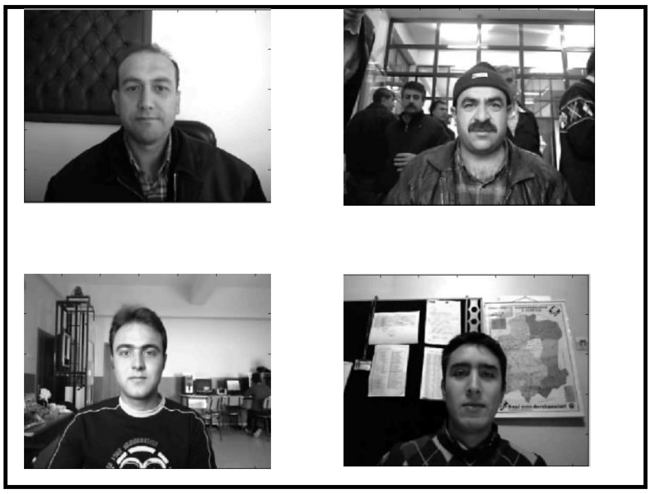

Figure 1: Four images from the database

The developed face detection system is based on a multi layer perceptron neural network (MLP) and written in Matlab. Only gray level images are being processed since our database contains only such images. A complete detection process of a human face in an image file is accomplished in two stages. In the first stage, a pre-defined face template is used to perform a search in the image in order to detect a right side of a human face. If an object similar to the template is found, the second stage of the detection algorithm takes place. In the second stage, another face template for the left side of the face is used to determine whether the encountered object is a human face or not. Block diagram of the detection method is given in the Figure 2, followed by detailed explanation of the detection process.

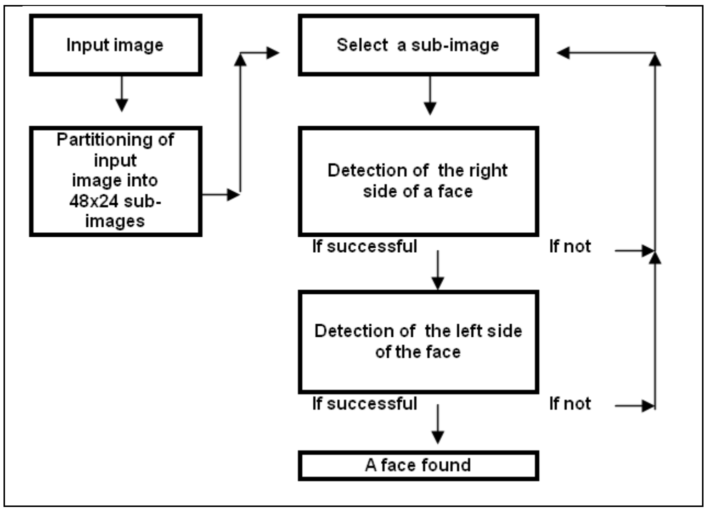
Input image
Partitioning of input image into 48x24 sub-images
Select a sub-image
Detection of the right side of a face
If successful
If not
Detection of the left side of the face
If successful
If not
A face found

Figure 2: Block diagram of the face detection process

As it is seen from the figure, the detection process starts with partitioning an input image into 48x24 sub-images. Each of these sub-images is fed into a detection algorithm in order to be classified as a face class or non-face class. Figure 3 shows some examples of these 48x24 sub-images.

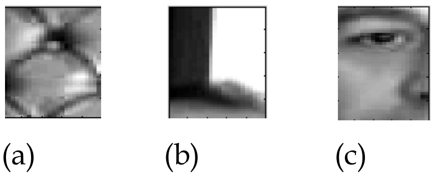

Figure 3: (a) and (b) Sample 48x24 sub-images of the non-face class. (c) A sample sub-image of the face class.

The MLP based detection algorithm used to classify sub-images is a two layer network: an input layer and an output layer. The input layer consists of **J** neurons (for our application **J** is set to 5) and the output layer contains 1 neuron. Figure 4 shows the block diagram of this network.

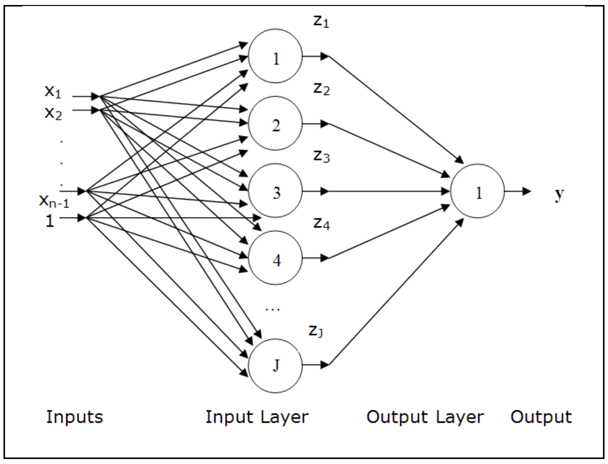

Figure 4: Block diagram of the 2 layer neural network used for detection of faces.

Our method detects a face in an image as follows: A sub-image selected, normalized and then converted to column format so to that it could be applied as an input to the network. The network classifies this sub-image as a face or non-face class. This process is repeated until a face class is detected.

When a face class is encountered, the network performs a second search by taking coordinates of the current sub-image frame as a reference point. The second search is also done by the same MLP but this time it uses the priori knowledge of that if current coordinates contain a half portion of a face, the other half should be located at the closed vicinity of the present frame. If the left portion of the face is found as well during the second search, the network identifies this coordinate pair as a location of a face at the output image.

The identification of sub-images by the network is performed with sequentially updated gradient descent algorithm (Hassoun, 1995; Zurada, 1992). The training of the above two layer perceptron network is done according to the procedure given below. Let's define

$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ 1 \end{bmatrix}_{nx1}$ $\epsilon\, \mathbb{R}^n$ as an input vector, $\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_J \end{bmatrix}_{Jx1}$ $\epsilon \mathbb{R}^J$, as a hidden layer output vector, $y\, \epsilon\, \mathbb{R}$ as an output of the network, $d\, \epsilon\, \mathbb{R}$ as a target for the output of the network,

$\mathcal{V} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ & & \dots & \\ v_{J1} & v_{J2} & \dots & v_{Jn} \end{bmatrix}_{Jxn}$ as weights of the input layer of the network and $\mathcal{W} = [w_{11} \;\; w_{12} \;\; \dots \;\; w_{1J}]_{1xJ}$ as weights of the output layer. For the input layer, we define the output of each perceptron as:

$$net_{z_1} = v_{11}.x_1 + v_{12}.x_2 + \cdots + v_{1\,n-1}.x_{n-1} + v_{1n}.1 \qquad (1)$$

$$z_1 = f_i\left(net_{z_1}\right)$$

(2) $z_J = v_{J1}.x_1 + v_{12}.x_2 + \cdots + v_{1\,n-1}.x_{n-1} + v_{1n}.1$

...

$$z_J = f_i\left(net_{z_J}\right) \qquad (3)$$

Where in vector form:

$$\boldsymbol{z} = \Gamma_i\left(net_{z1}, net_{z2}, \dots, net_{zJ}\right) = \Gamma_i(\,\mathcal{V}.\boldsymbol{x}\,) = \begin{bmatrix} f_i(net_{z1}) \\ f_i(net_{z2}) \\ \dots \\ f_i(net_{zJ}) \end{bmatrix} \qquad (4)$$

The $f_i(net)$ function for the input layer is used as:

$$f_i(net) = \frac{e^{\beta.net} - e^{-\beta.net}}{e^{\beta.net} + e^{-\beta.net}}$$

Similarly, the perceptron output of the output layer is defined as:

$$net_o = w_{11}.z_1 + w_{12}.z_2 + \cdots + w_{1J}.z_J \qquad (5)$$

$$y = f_o(net_o) = \Gamma_o(\,\mathcal{W}.\boldsymbol{z}\,) = \Gamma_o\left(\,\mathcal{W}.\Gamma_i(\,\mathcal{V}.\boldsymbol{x}\,)\right) \qquad (6)$$

Where $f_o(net)$ is

$$f_o(net) = \frac{e^{\alpha.net} - e^{-\alpha.net}}{e^{\alpha.net} + e^{-\alpha.net}}$$

For the above network the statement of the learning problem can be given as follows: Given a training set of $\{(\boldsymbol{x}^1, d^1), (\boldsymbol{x}^2, d^2), \dots, (\boldsymbol{x}^m, d^m), \dots\}$ for m = 1, 2,…, M , find $\mathcal{V}^* and\ \mathcal{W}^*$ such that all output y^m are as close as corresponding targets d^m.

The gradient descent algorithm is used to train the network. The algorithm is based on computing current error function for $(\boldsymbol{x}^k, d^k)$ at any step **k** and moving the weights of input and output layers of the network towards the direction of negative gradient along the error surface. The error function at training step **k** is chosen as

$$E^k(\mathcal{V}, \mathcal{W}) = \frac{1}{2}(d^k - y^k)^2 = \frac{1}{2}\left(d^k - f_o(net_o)\right)^2 \qquad (7)$$

The weights of the input layer are updated according to following equation:

$$v_{ji}^{k+1} = v_{ji}^k - \rho_v . \nabla E^k(\mathcal{V}, \mathcal{W}) \qquad (8)$$

and for the output layer, the updating of the weights is done as:

$$w_{lj}^{k+1} = w_{lj}^k - \rho_w . \frac{\partial E^k(\mathcal{V}, \mathcal{W})}{\partial w_{lj}} \qquad (9)$$

In order to find exact expressions for both weight functions, we will compute gradients using the Chain rule starting with weights of the output layer.

$$\frac{\partial E^k(\mathcal{V}, \mathcal{W})}{\partial w_{lj}} = \frac{\partial E^k(\mathcal{V}, \mathcal{W})}{\partial y^k} . \frac{\partial y^k}{\partial net_o^k} . \frac{\partial net_o^k}{\partial w_{lj}} \qquad (10)$$

$$\frac{\partial E^k(\mathcal{V}, \mathcal{W})}{\partial y^k} = -1.(d^k - y^k) \qquad (11)$$

$$\frac{\partial y^k}{\partial net_o^k} = \frac{\partial f_o(net_o^k)}{\partial net_o^k} = f_0^{'}(net_o^k) \qquad (12)$$

$$\frac{\partial net_o^k}{\partial w_{lj}} = z_j \qquad (13)$$

$$w_{lj}^{k+1} = w_{lj}^{k} + \rho_w . (d^k - y^k) . f_0^{'}(net_o^k) . z_j \quad (14)$$

For the output layer:

$$\frac{\partial E^k(\mathcal{V},\mathcal{W})}{\partial v_{ji}} = \frac{\partial E^k(\mathcal{V},\mathcal{W})}{\partial z_j} . \frac{\partial z_j}{\partial net_j^k} . \frac{\partial net_j^k}{\partial v_{ji}} \quad (15)$$

$$\frac{\partial E^k(\mathcal{V},\mathcal{W})}{\partial z_j} = -1 . (d^k - y^k) f_0^{'}(net_o^k) . w_{lj} \quad (16)$$

$$\frac{\partial z_j}{\partial net_j^k} = \frac{\partial f_i\left(net_j^k\right)}{\partial net_j^k} = f_i^{'}\left(net_i^k\right)$$

(17)

$$\frac{\partial net_j^k}{\partial v_{ji}} = x_i \quad (18)$$

$$v_{ji}^{k+1} = v_{ji}^{k} + \rho_v . (d^k - y^k) f_0^{'}(net_o^k) . w_{lj} . f_i^{'}\left(net_i^k\right) . x_i \quad (19)$$

We have implemented the above training algorithms with Matlab. In order to train the MLP network, we have selected 25 people from the database to form a training set. For each person, we have obtained 10 sample image frames of 48x24 that correspond to the face of that individual from little above the eye browses to the mouth. Five of these 10 samples are obtained from some close coordinates at little above the left eye and the other five are obtained from similar coordinates around the right eye. Total of 250 samples are collected that would constitute the face class of the training set. To collect non-face class samples of the training set, we have selected 3 images from the training set which are obtained with different backgrounds. We then sampled these 3 images at locations that did not correspond to face regions in order to get 758 separate 48x24 sub-images. Hence the total number of image frames in the training set was 1008.

Our network successfully minimized the error function during the training phase. The Figure 5 shows plot of Error (Energy) function as function of number of iterations on the training set during training.

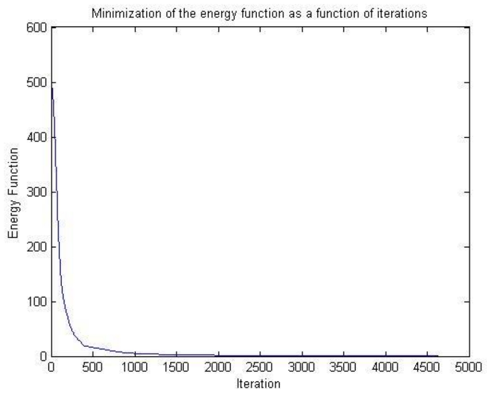

Figure 5: Change in energy function during training**.**

3. RESULTS

We have tested our method on a dataset consisted of 100 people selected form our database. For each image in the test set, the detection program generates an output image as the same size as the input image. This output image contains the numerical values of the output of the neural network for the cases where the network detects a face for those locations. The remaining locations of the output image contain values of zero.

Simulation results are presented on the basis whether a face in a test image is detected by the method or not. This is accomplished by selecting a threshold value. If the output of the network exceeds that threshold at any location of the input image, it is considered that there is a face there. Then, the value of the output of the network which is a real value between -1 and 1 is written to the same location of the output image. Here, a value close to 1 indicates a location of possible correct face detection. Since we are only interested in values close to 1 at the output, values less than zero are replaced with zeros in output images.

Table 1 shows simulation results. The first column indicates the time needed to train the network. The second column shows the total amount of time for the method to detect faces for 100 images in the test set. The third column gives the correct detection rate of the method. When calculating the detection rate, we proceed as follow: if the network finds a face correctly in a test image, it is considered as a correct detection regardless of any other wrong detection in the same test image. If the network does not find a face in a test image at all, this case is taken as a wrong detection.

Table 1: Simulation results.

Training Time (Second)	**Detection Time (Second)**	**Detection Rate %**
651	314	81

When we look at output images that face portion is detected successfully, we see two different cases. In the first case, there are some output images where the method detects the face (or faces) in the image correctly and no other places in the output image are marked as faces. In the second case, there are some output images where the method correctly classifies the face portion but some other locations in the input image are also marked as faces while those locations do not actually correspond to a face on an individual. The Figure 6 shows sample output images of these two cases.

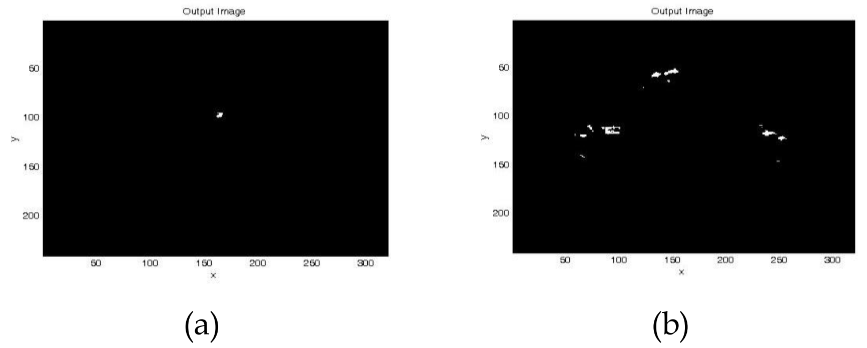

Figure 6: a) An output image where a face is found at only one place correctly b) An output image where correct and wrong detections accrued simultaneously.

After performing couple of experiments, we have found that the network performs wrong detection because there are not sufficient numbers of data points in the training set representing non-face class. However, when we add more number of non-face data to the training set, data points in the training set becomes inseparable, i.e. the error function does not decrease during the training. We will perform some further study to solve this problem in our next work.

4. CONCLUSION AND FUTURE WORKS

In this study, we have developed a method based on a multilayer perceptron network in order to detect faces in gray scale images. The method is tested on a database of 100 images each belonging to a different individual. The method performed correct detection rate of %81 on this data set. However, for the majority of the cases that the network found the correct location of the face, it had also performed a number of wrong detections in the same test image. We will address this problem in our next study.

In our future work, we will study on this subject further in order to increase our method's correct detection capability.

REFERENCES

İşler, Y. S., Salihmuhsin, M. (2008). Bir Tek Nöron Kullanılarak Resimler İçerisinde Göz Kısmının Bulunması. Kahramanmaras Sutcu Imam University Journal of Engineering Sciences. Volume 11-1, pp: 59-63.

Direkoglu, C., Demirel, H., Ozkaramanli, H., Uyguroglu, M. (2005). Efficient Face and Facial Feature Tracking Using Search Region Estimation. Image Analysis and Recognition, LNCS, Vol: 3656, pp: 1149 – 1157.

Yacoub, B., Fasel, B., Lüttin, J. (1999). Fast face detection using MLP and FFT. Proceedings Second International Conference on Audio and Video based Biometric Person Authentication (AVBPA).

Nanni, L., Lumini, A. (2007). A multi-expert approach for wavelet-based face detection. Pattern Recognition Letters,Volume 28 , Issue 12, pp 1541-1547.

Lin, S. H., Kung, S. Y., Lin, L. J. (1997). Face Recognition/Detection by Probabilistic Decision Based Neural Network. IEEE Transaction on Neural Networks, Volume 8, No: 1, pp 114-132.

Moghaddam, B., Pentland, A. (1997). Probabilistic Visual Learning for Object Recognition. IEEE Transaction on Pattern Recognition and Machine Intelligence, Volume 19, No: 7, pp 696-710.

Cootes, T. F., Taylor, C. J. (1996). Locating Faces Using Statistical Feature Detectors. Proceeding of Second International Conference on Automatic Face and Gesture Recognition, pp 204-209.

Han, C. C., Liao, H. Y. M., Yu, K. C., Chen, L. H. (1998). Fast Face Detection via Morphology-Based Pre-Processing, Proceeding of Ninth International Conference on Image Analysis and Processing, pp 469-476.

Feraud, R., Bernier, O.J., Villet, J. E., Collobert, M. (2001). A Fast and Accurate Face Detector Based on Neural Networks. IEEE Transaction on Pattern Recognition and Machine Intelligence, Volume 22, No: 1, pp 42-53.

Rowley, H., Baluja, S., Kanade, T. (1996). Neural Network-Based Face Detection. Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp 203-208.

Yang, M. H., Kriegman, D. J., Ahuja, N. (2002). Detecting Faces in Images: A Survey. IEEE Transaction on Pattern Analysis and Machine Intelligence. Vol 24, No: 1, pp 34-58.

Hassoun, M. H. (1995). Fundamentals of Artificial Neural Networks, The MIT Press, Massachusetts.

Zurada, J. M. (1992). Introduction to Artificial Neural Networks, West Publishing Company.