Contemporary Engineering Sciences, Vol. 4, 2011, no. 4, 177 – 186

A Non-linear Supervised ANN Algorithm for Face Recognition Model Using Delphi Languages

Mahmood K. Jasim[1]

DMPS, College of Arts & Sciences, University of Nizwa, OMAN
mahmoodkhalid@unizwa.edu.om

Ahmad Fouad Alwan

Electronic Dept., Technical Industrial Institute-Haddah
Sana'a, Republic of Yemen
afa1956@yahoo.com

A. M. A Dawagreh

Chemical Engineering Dept., Al-Huson University College
Al-Balqa Applied University, Jordan
adawagreh@yahoo.com

Muna Abdul Husain Radhi[2]

Computer Sciences Dept., College of Sciences
Al Mustansiryha University, Iraq
muna.radhi2000@yahoo.com

Abstract

Artificial neural networks (ANN) have been developed in a wide variety of configurations and represent a major extension of computation. The theoretical foundations of artificial neural networks are expanding rapidly, but they are currently inadequate to support the more optimistic projections. ANN has been

[1]&2 Corresponding authors

used to recognize images of human-being face. An algorithm was designed on the basis of using back-propagation, pattern recognition and image processing. The algorithm so constructed was applied on different images of human-being faces were implemented using Delphi language version 5 for different poses. The results so obtained can recognize the face of a given image regardless of the pose.

Keywords: Face recognition, Artificial Neural Network, Mathematical Modeling

1- Introduction

Face and facial expression recognition have attracted much attention though psycho-physicists, neuroscientists, and engineers have studied them for more than 20 years. A first step of any face processing system is detecting the locations in image where faces are present. However, face detection from a single image is a challenging task because of variability in scale, location orientation (up-right, rotated), and pose (frontal, profile) facial expression, occlusion, and lighting condition also change the overall appearance of faces [1]. The challenges associated with face detection can be attributed to the pose, Presence or absence of structural components, Facial expression, Occlusion, and Image orientation & conditions [2]. The goal of facial feature detection is to detect the presence and location of features, such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc., with assumption that there is only one face in an image [3], [4]. Face recognition (identification) compares an input image against a database and report a match, if any [5], [6], [7], [9].

Neural networks have been applied successfully in many pattern recognition problems, such as optical character recognition, object recognition and autonomous robot driving. In the present paper, our interest may goes through the following neural architecture to deal with nonlinear supervised case to be studied.

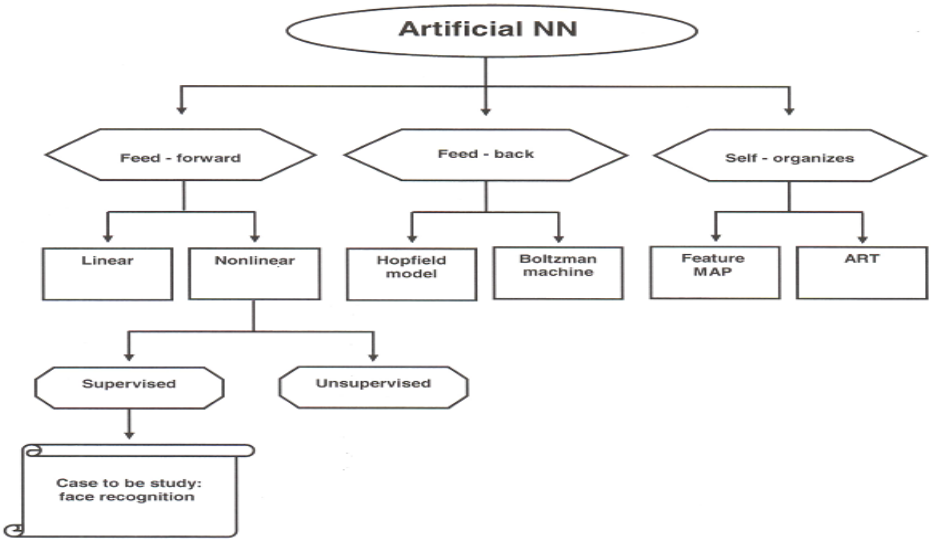
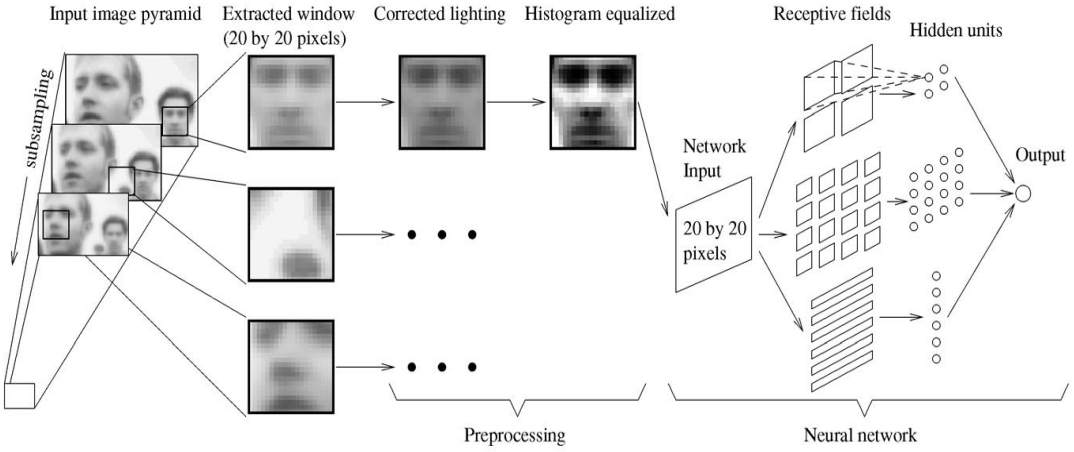


Figure (1), shows the architecture of ANN

2- Basic concepts

Neural networks (NN) are developed with the goal of modeling information processing and learning in the brain, this can be applied to a number of practical applications in various fields. The advantage of using NN for face detection is the feasibility of training system to capture the complex class conditional density of patterns. However, one drawback is that the network architecture has to be extensively turned to get exceptional performance. An early method using hierarchical NN was proposed by Agui et al [10]. Among all the faces detection methods that used neural networks, the most significant work is done by Rowley et al [11] as in figure below.



The challenges associated with face detection can be attributed to the following factors [8]:

- Pose
- Presence or absence of structural component
- Facial expression
- Occlusion
- Image orientation
- Imaging conditions

The back-propagation supervised learning algorithm is used to find weights in multilayer feed forward networks. The errors resulting from the comparison of actual and target output values are propagated backward through the network, and weight values are adjusted to minimize error. As long as the network continues to generate values closer to the validation values, training continues. The training simulation process ends when all patterns are classified correctly within a selected range of accuracy, or the network begins to perform more poorly in a process called over-fitting or over-training and if the network is unable to learn all the patterns, the network topology can be adjusted [9].

With concern to neural network, the performance criterion is the minimization of squared error. Therefore, the total system error (*E*) is expressed as follows:

$$E = \sum_{p,i} \left(t_{ip} - y_{ip}\right)^2$$

Where i indexes units of output; p indexes the input-output pairs to be learned; t_{ip} refer to the desired output, and y_{ip} is the network's calculated output. This function is minimized, and if the output functions are differentiable, the task of blame assignment is simplified. Most of the neural network aims are to train the network in order to achieve a balance between the ability to respond correctly the input patterns that are used for training (memorization) and the ability of giving reasonable but not identical to that used in training (generalization).

3- A training algorithm by back-propagation using Delphi

Once the network layout and computational characteristics of the network are established, the network's adaptive learning. Neural networks must, learn, to generate values consistent with the patterns in a given data set to make accurate projections. The learning process is when network weights change in response to a training data set of artificial neural network (ANN)'s learn in two ways: supervised and unsupervised learning, which differ according to whether or not known answers are used to train the network.
Supervised training is used when the data set contains target output values associated with and minimizes the errors by discovering the driving features in the data and adjusting the weights.
Once trained, the network will be able to discover patterns and make predications with data not used in the training set. The most common algorithm used adjusting the weight in supervised training is called back-propagation. The training of a network by back-propagation involves three stages:
1. The feed forward of the input training pattern.
2. The calculation and back-propagation of the associated error.
3. The adjustment of the weights.

-During feed forward each input unit (X) receives an input signal and broadcasts this signal to the each of the hidden units ($Z_I ... Z_P$). Each hidden unit then computes its activation and sends its signal (Z_I) to each output unit. Each output units (Y_k) computes its activation (Y_k) to form the response of the net for the given input pattern.

-During training each output unit compare its computed activation Y_k with its target value t_k to determine the associated error for that pattern with that unit. Based on this error the factor $\delta_k \left(K = 1..m\right)$ is computed. δ_k is used to distribute

the error at output unit Y_K back to all units in the previous layer (the hidden units that are connected to Y_K).it is also used (layer) to update the weights between the output &hidden layer.

In a similar manner, the factor δ_j $(j = 1...p)$ is necessary to propagate the error back to the input layer but δ_j is used to update the weights between the hidden layer and the input layer.

After all of the δ factors have been determined the weights for all layers are adjusted simultaneously. The adjustment to the weight w_{ik} (from hidden units z_j to output unit Y_k) is based on the factor δ_k and the activation Z_j of the hidden units Z_i. The adjustment to the weight v_{ij} (from input unit X_i to hidden unit Z_j) is based on the factor δ_j and the activation X_i of the input unit. So, the following algorithms have been constructed as a training algorithm:

Step 0: Initialize weights
Step 1: while stopping condition is false .do steps 2-9.
Step 2: for each training pair. Do step 3-8.

Step 3 & step 4 {feed forward}

Each input unit ($X_i, i = i...m$) receives input signal X_I and broadest this signal to all units in the layer above and each hidden units $(Z_j, j = 1...p)$ sum its weighted input signals $Z\_in_j = v_0 + \sum_{i=1}^{n} X_i V_{ij}$ applies its activation function to compute its output signal. $Z_j = f(Z\_in_j)$ follows by sending this signal to all units in the layer above.

Step 5:

Each output unit $(Y_k, k = 1...m)$ sums its weighted to input signals and applies its activation function to compute its output signal

Step 6: {back - propagation of error}

Each output unit $(Y_k, k = 1...m)$ receives a target pattern cores pending to the input training pattern.

- Compute its error information term $\delta_k = (t_k - Y_k) f^{-}(Y\_in)$
- Calculates its weight correction term (used to update w_{ik}) $\Delta w_{ik} = \alpha . \delta_k . z_j$.
- Calculates its bias correction term (used to update w_{0k}) $\Delta w_{0k} = \alpha . \delta_k$ and send δ_k to units in the layer below.

Step 7:

Each hidden units $(Z_j, j = 1...p)$ sums is delta inputs (from units in the layer above). $\delta\_in_J = \sum_{k=1}^{m} \delta k.wjk$ multiplies by the derivative of its activation function to calculate its error information term $\delta_j = \delta\_in_j . f^{-}(Z\_in_j)$

Calculates its weight correction term (used to update v_{ij}) $\Delta v_{ij} = \alpha \ . \ \delta_j . X_i$ and calculate its bias correction term (used to update v_0) $\Delta v_{0i} == \alpha \ . \ \delta_j$.

Step 8: {up date weights & biases}

- Each output unit $(Y_k, k = 1...m)$ updates its bias and weights $j = 0...p$, $w_{ik}(new) = w_{ik}(old) + \Delta w_{ik}$
- Each hidden unit $(Z_j, j = 1...p)$ updates its bias and weights $(i = 0...n)$ $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

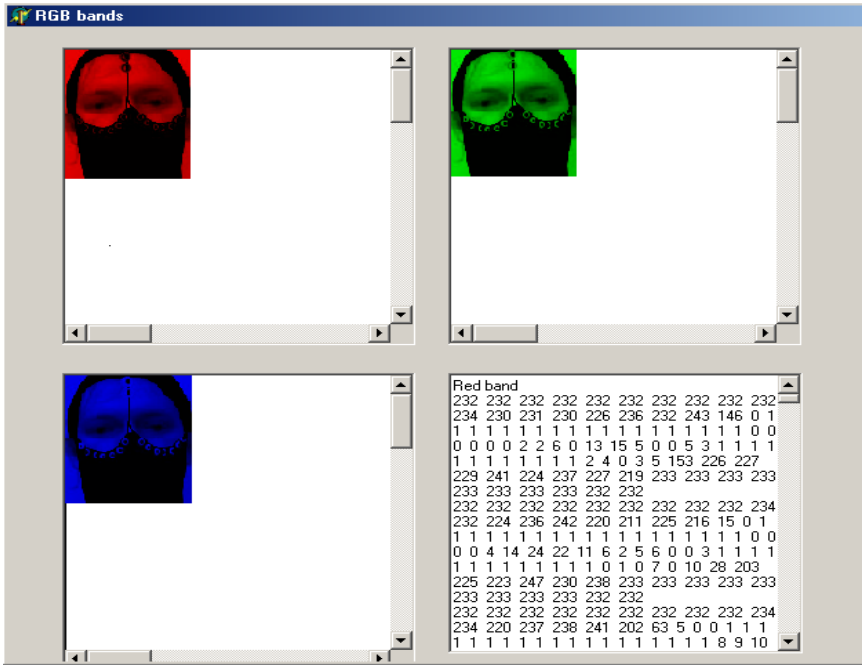
Step 9: Test stopping condition.

4- Simulation Results

It is very difficult to know which training algorithm will be fastest for a given problem. It is depends on many factors, including the complexity of the problem, the number of data in the training sets, weights and biases in the network, and the error. The best error performance with significant reduction has been consumed in the training and the procedure for running the algorithm may goes through the following:

- Open the image
- Separate the image in three bands (red, green, blue) and store each bound in array.
- Indicate distinguish areas from the image (eyes, nose, and mouth) and separate it in different block (16*16) pixel to every image for the same person

Thus, the following screen shows the matrix of the band to the face that has been studied.



Now in order to learn the network to different blocks for different image of the person using back-propagation with Delphi languages version 5

Back-propagation

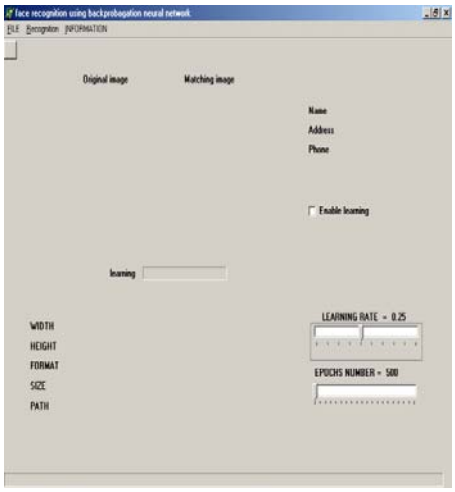
- Learn the net to different blocks for different image of the same person

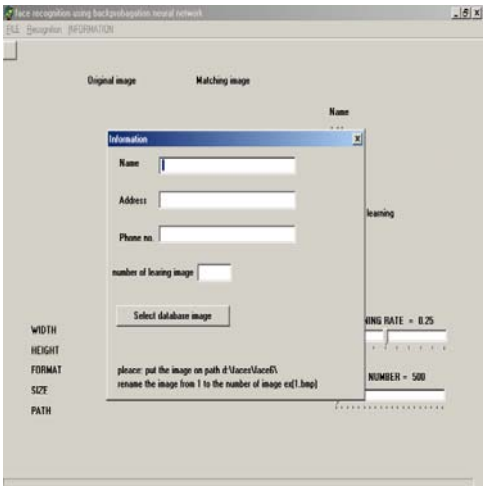
Net properties

1. Input node 256 nodes (16*16).
2. Hidden node 64 node
3. Output node 256 node (represent the no. of image in database).

- After learning the set of image we store the weight (w,v) in separate file and store the information of the person and the base image.
- Repeat the same previous step for all sets of image for the same person.

All such processes may give arise through the flowing screens:

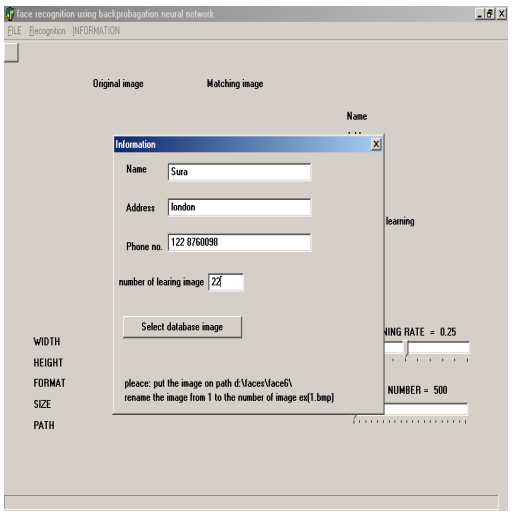


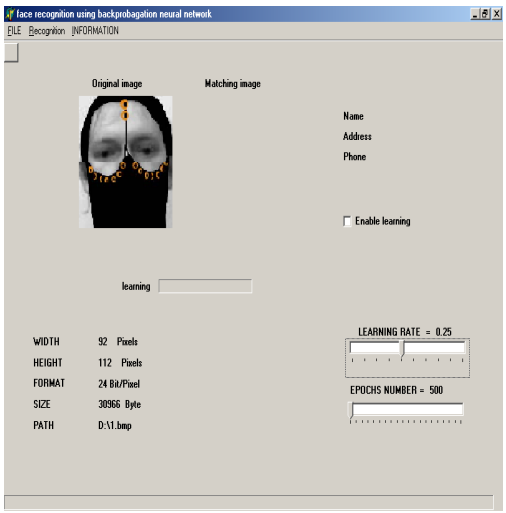


In order to recognition, the faces:
Recognition

- Retrieve the weight (w,v).
- Enter the image to the net and find the output (which represents the sequence of image in database).
- Load the image and information and picture from database.

The following screens shows the procedure of matching the original image with others





5. Conclusion

Summing up for face recognition the following have been concluding
Face Recognition: Training Data

- Classifying photo images of faces of people in various poses through the direction (left, right, straight ahead, up) and person's expression (happy, sad, angry, and neutral).
- Data

84 Grayscale images for 5 different people using 20 Images per person, varying

- Resolution of images: 92*112, each pixel with a grayscale intensity between 0 (black) and 255 (white)

Face Recognition: Factors for ANN Design

- Input encoding : Images
- Output encoding : number of image in data base

Face Recognition: Input encoding

- Possible Solutions

Extract key features using preprocessing

- Features extraction: Edges, region of uniform intensity

Face Recognition: output encoding

- Possible coding schemes by using multiple output unit with single threshold value

Face Recognition: Network Structure

Input nodes: 256
Hidden nodes: 64
Output nodes: 256

Face Recognition: Other Parameters

- Learning rate $\eta = 0.5$
- Weight initialization: small random values between (1,-1)

- Number of iteration (500 to10000).

Acknowledgments
The authors wish to express their sincere thanks to the referees, Editorial Board, as well as to Dr. Emil Minchev President of Hikari Ltd Managing Editor of Contemporary Engineering Sciences Journal.

References

[1] K Lam and H Yan, "Fast Algorithm for Locating head Boundaries", J Electronic Imaging, Vol 3 No 4 PP 351-359, (1994)
[2] B Monghaddam and A Pentland, " Probabilistic Visual Learning for Object Recognition", IEEE Trans-Pattern Analysis and Machine Intelligence", Vol 19, No 7 PP 696-710, (1997)
[3] I Craw, D Tock, and A Bennet, "Finding Face Features", Poc. Second European Conf., Computer vision, PP 92-96, (1992)
[4] H P Graf, T Chen, E Petajan and E Costatto," Locating Faces and Facial Parts", Proc. 1st Int'l Workshop Automatic Face and Gesture Recognition, PP 41-46, (1995)
[5] M Turk and A Pentland, "Eigenfaces for Recognition", J Cognitive Neuroscience, Vol 141, pp 245-250, (1994)
[6] A Samal and P A Iyengar," Automatic Recognition and Analysis of Human Faces and Facial Expressions: A Survey", Pattern Recognition, Vol 25, No 1 PP 65-77, (1992)
[7] R Chellappa, C L Wilson and S Sirohey," Human and Machine Recognition of Faces: A Survey", Proc. IEEE Vol 83, No 5, PP 705-740, (1995)
[8] Ming-Hsuan Yang, David J Kriegman and Narendra Ahuja, " Detecting Faces in Images: Survey", IEEE Trans on Pattern Analysis and Machine Intelligence, Vol 24, No 1, (2002)
[9] M K Jasim and Muna A Radhi, "On artificial neural network for face recognition application", College of Education Journal, Al Mustansiryha Univ., Vol. 2, (2006)
[10] T Agui, Y Kokubo, H Nagashashi and T Nagao, "Extraction of Face Recognition from Monochromatic photographic using NN", Proc. 2nd Int'l Conf. Automation
Robotics and computer vision, Vol 1, (1992)
[11] H Rowley, S Baluja and T Kanade, "Neural network based face Detection", IEEE Trans pattern Analysis and Machine Intelligence Vol 20, No 1 pp 23-38, (1998)

Received: February, 2011