MOUAYED ENATTAH

TRAFFIC SIGN RECOGNITION BASED ON
ARTIFICIAL NEURAL NETWORK

NEU
2017

TRAFFIC SIGN RECOGNITION BASED ON ARTIFICIAL NEURAL NETWORK

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
MOUAYED ENATTAH

In Partial Fulfilment of the Requirements for The Degree of Master of Science in Electrical and Electronic Engineering

NICOSIA, 2017

TRAFFIC SIGN RECOGNITION BASED ON ARTIFICIAL NEURAL NETWORK

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
MOUAYED ENATTAH

In Partial Fulfilment of the Requirements for the Degree of Master of Science in Electrical and Electronic Engineering

NICOSIA, 2017

MOUAYED ENATTAH: TRAFFIC SIGN RECOGNITION BASED ON ARTIFICIAL NEURAL NETWORK

Approval of Director of Graduate School
of Applied Sciences

Prof. Dr. Nadire ÇAVUŞ

We certify this thesis is satisfactory for the award of the degree of Masters of Science in Electrical and Electronic Engineering

Examining Committee in Charge:

Assist.Prof.Dr. Kamil Dimililer	Department of Automative Engineering, NEU
Assist.Prof.Dr. Samet Biricik	Department of Electrical and Electronic Engineering, EUL
Assist.Prof.Dr. Boran Şekeroğlu	Department of Information Systems Engineering, NEU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Mouayed Enattah

Signature:

Date:

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and thanks to my beloved supervisor, Assist. Prof. Dr. Boran Sekeroglu , for his continuous guidance and assistance during my graduate studies. His inspiring knowledge and creative thinking have been source of encouragement throughout this work. My deepest gratitude goes to my brothers, sisters, and to whom I am most indebted. I thank them for the love, prayers, patience and support while I was studying. I know I can never come close to returning their favour upon me. I will always be thankful to my friends and colleagues for their unlimited support. I extend my thanks to all the Libyan community that gave me a second family away from home.

To my parents and family….

ABSTRACT

The spread of the use of driver assisting system in the recent years and the spread of smart systems is increasing. These systems investigate in the computer algorithms and sensor accessories to offer safer and luxurious driving experience in the modern cars. Autonomous vehicles are also being investigated and implemented to be driven without a driver. In order for these systems to be successful and safe, they need to be able to understand all traffic signs in the road and be able to react to these signs. Different researches have been presented discussing the implementation of algorithms for traffic signs recognition. The use of artificial neural networks is also proving high capability and offering good performance. This work discusses the implementation of a neural network based traffic sign recognition system. The proposed study uses 13 different traffic signs to train and test the artificial neural network. The system also implements a segmentation algorithm to study its effect on the performance of the artificial neural network. Different parameters were used during the training of the neural network. The results of the system implementation were obtained, tabulated, and discussed.

***Keywords*:** Artificial neural networks; back propagation; recognition; traffic sign; image processing

ÖZET

Yakın zamanda sürücü destek sistemleri ve akıllı sistemlerin kullanımı yaygınlaşmaktadır. Bu sistemler bilgisayar algoritmalarını ve sensor aksesuarlarını inceleyerek modern arabalarda daha güvenli ve konforlu sürüş deneyimi sunmaya çalışmaktadır. Özerk araçlar da sürücüsüz kullanım için araştırılmakta ve uygulanmaktadır. Bu sistemlerin güvenli ve başarılı olabilmeleri için, yollardaki trafik işaretlerini anlamaları ve bu işaretlere reaksiyon vermeleri gerekmektedir. Trafik işaretlerinin tanımlanması uygulaması için bir çok farklı araştırma sunulmuştur. Yapay Sinir Ağları'nın kullanımı da trafik işaretlerinin algılanması ve tanımlanmasında yüksek kapasite ve yüksek performans sağlamaktadır. Bu çalışmada, yapay sinir ağları tabanlı trafik işaretleri tanımlaması uygulaması sunulmaktadır. Çalışmada yapay sinir ağlarının öğrenimi ve testi için 13 farklı trafik işareti kullanılmıştır. Sistem ayrıca segmentasyon algoritması kullanarak yapay sinir ağlarının performansının ölçülmesini içermektedir. Yapay sinir ağlarının eğitimi sırasında birçok farklı parametre kullanılmıştır. Uygulanan systemin test sonuçları elde edilmiş, değerlendirilmiş ve tartışılmıştır.

Anahtar Kelimeler: Yapay Sinir Ağları; Geri Yayılmalı ağlar, tanımlama; trafik işaretleri; görüntü işleme

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN:	Artificial Neural Networks
BPANN:	Back Propagation Artificial Neural Networks
LR:	Learning rate
MF:	Momentum Factor
MSE:	Mean Squared Error

CHAPTER 1
INTRODUCTION

1.1 Introduction

Traffic signs have very great importance in creating a secure travel atmosphere and control the flow of traffic all around the world. Traffic signs provide drivers with great amount of information about the limitations, directions, quality of streets, possible dangers, and help drivers to drive in an ordered manner. The existence of traffic systems and signs is very important in the creation of safer world (Kobayashi, Baba, Ohtani, & Li, 2015). The traffic system signs contain mainly three different shapes to simplify the task of recognition for drivers. These are the circular shape, the triangular shape in addition to some hexagonal and squared traffic signs. Each sign has its meaning and gives the drivers some information and instructions about the state of the road.

The recognition of traffic signs in streets is very important and becomes one of the challenges in modern technologies. Human drivers are able to detect, analyse, identify, classify, and recognize the different traffic signs with least effort and super excellent performance. This is not the case for automated computerized systems where all these tasks are considered very complicated and difficult to be carried out. The identification of traffic signs using computerised processes is very difficult due to variations in the illumination levels during the day and night, blurs due to motion, colour noise, occlusion and many other reasons (Yang, Luo, Xu, & FuchaoWu, 2016).

The development of self driven cars and the world tendency toward more use of smart technologies in addition to the increasing needs for safety in the roads encouraged researchers to focus more and more on automated recognition of traffic signs under different conditions. Modern smart cars need to be able to detect the different traffic signs to decide how to react for these signs. The development of such systems has become a social demand (Kobayashi et al., 2015). The automatic recognition of traffic signs will help to increase the safety in streets as such a system can e used to alarm the driver about possible danger he might not be aware of. They also can be used to provide smart cars with additional information concerning the road.

Artificial neural networks were used widely in classification and recognition tasks of different types. They are implemented in person identification and recognition based on faces images, fingerprint, ear print, and eye retina. Neural networks are also used in text recognition, signature recognition, disease identification, and many other applications. Artificial networks have proven their ability to perform complicated tasks based on their brain like structure. They imitate the structure and function of human brain. They are constructed of interconnected neurons arranged in different layers and pass the information between each other to carry out specific tasks.

In this work, the artificial neural networks are used in the recognition of different traffic signs based on their images. 13 different traffic signs were chosen to be implemented in the recognition process. Each signs has 16 different images created using different types of noise and effects. The images were edited simply using paint program and all used for the training and test of the neural network structure. The training task of the network was carried out based on the well known back propagation algorithm.

CHAPTER 2

IMAGE PROCESSING TECHNIQUES

2.1 Introduction

Image processing techniques have been used and implemented widely in the field of classification and recognition. Digital image processing has become very famous in engineering, medical sciences, computer sciences, and information technologies. Different researches and activities in commercial and non commercial perspectives have revealed the importance of this science in the last years. This science, like most of other modern sciences, has gained its importance as a result of the development of digital processing systems and silicon technologies. Three main categories can be found in image processing, these are coding of the image, analyzing the image, and restoration enhancement of the image. Each one of these three categories has its own separate methods and goals. The image analysis has as purpose the extraction of quantitative measures of the image aspects. The enhancement restoration's purpose is the rejection of undesired features caused by different types of noise or errors.

This chapter will discuss some of the image processing techniques that can be useful in the achievement of this thesis goal by helping in the process of classification of the traffic signs.

2.2 Weiner Filter

The Weiner filter can be described as a frequency domain based weighting function. It is based on the assumption that the energy of spectral density of the noise signal is higher at high frequency. The filter has better attenuation if the spectral density of a signal is smaller. Wiener filter is an optimum filter that is used for linear estimation of wanted signal. In wiener filter approach, the signal S_x is assumed to be noised by the signal S_n. the noise signal is considered to be zero mean and having a standard deviation $\boldsymbol{\delta}$. The application of wiener filter will produce a pure signal (free of noise), wiener filter is given by:

$$G(f) = \frac{S_x}{S_x + S_n} \qquad (2.1)$$

The optimization idea is to reduce an error function (generally mean squared error) by using filter adaptation. This solution is known by wiener filtering. The wiener filter can be described using the block diagram of the next figure. Convolution is applied between the random input signal W and the Wiener filter transfer function G. the output result is subtracted from the reference S. The result of subtraction is the error signal used to adjust the Weiner filter .

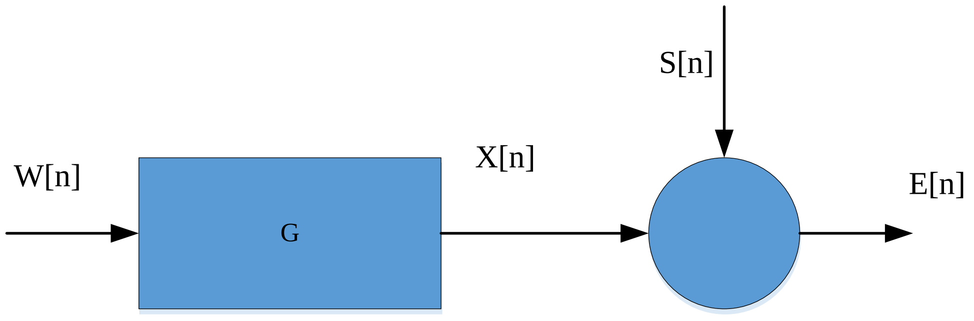


Figure 2.1: Block diagram of described wiener filter

Wiener filter is a causal filter type that is based on the assumption of the signal and the noise. It uses the minimum squared error as convergence criteria. Wiener filter offers good noise removal results as it implements statistical data of noise field in the filtering process. Based on statistical data of image and noise, the transfer function of the filter is evaluated to guarantee a minimum mean squared error signal. Wiener filter is applied on window or small part of the image whose choice is function of the noise characteristics (Santra, 2013).

Wiener filter implemented using MATLAB estimates the noise based on the variance and mean in the neighbourhood of each pixel. The mean value around each pixel inside a window defined by the length M and the width N is given by (Lim, 1990):

$$\mu = \frac{1}{M*N}\sum_{i=1}^{N}\sum_{j=1}^{M} a(i,j) \tag{2.2}$$

The variance of the window of neighbouring elements is defined by:

$$\sigma^2 = \frac{1}{M*N}\sum_{i=1}^{M}\sum_{j=1}^{N} a^2(i,j) - \mu^2 \tag{2.3}$$

The pixel wise estimation of the Wiener filter elements is done then using the equation:

$$W(i,j) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}(a(i,j) - \mu) \qquad (2.4)$$

The term v^2 refers to the variance of the noise signal. From the above equations, it is noticed that the performance of the Wiener filter is dependent on the estimation of the noise variance (Lim, 1990). Figure 2.2 represents the well known Lenna image before being noised. Gaussian noise that has zero mean and a small variance is going to be added to the image as presented in Figure 2.3. The application of adaptive wiener filter on the noisy image is presented in Figure 2.4. It shows how the filter has reduced the noise from the image based on the statistical analysis of pixel neighbouring window.

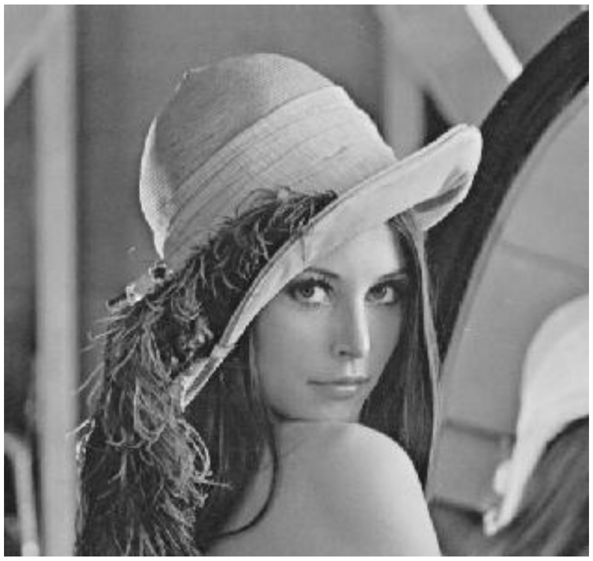


Figure 2.2: Lena image without noise

2.3 Median Filter

Median filtering is a technique in image processing that is used to decrease the level of undesired noise in an image. The median filter principle is based on the idea of removing all the strange pixels from a window of the image. This technique is classified within the non linear filtering techniques. The median filter is very well known for its performance in removing impulsive type noise that appears suddenly in some pixels. One of the important examples of this type of noise is the so called salt & pepper noise. Median filter is very efficient in removing this type of noise from images and restoring the original images with the minimum losses of pixels. The features of the filtered images are kept unchanged as it just replaces very unwanted different values by values that are in the neighbourhood region.

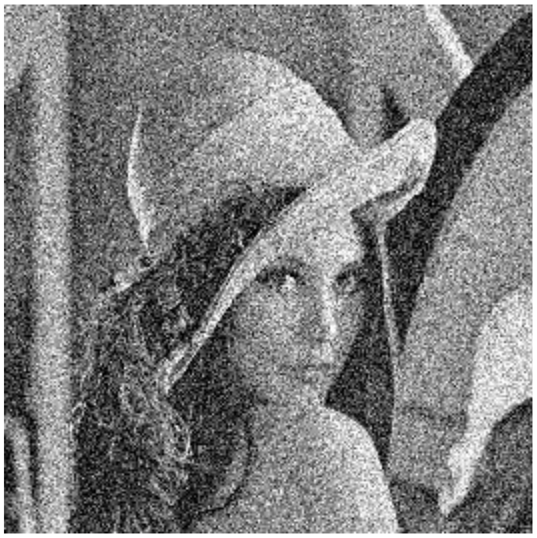


Figure 2.3: Lena image after adding Gaussian noise

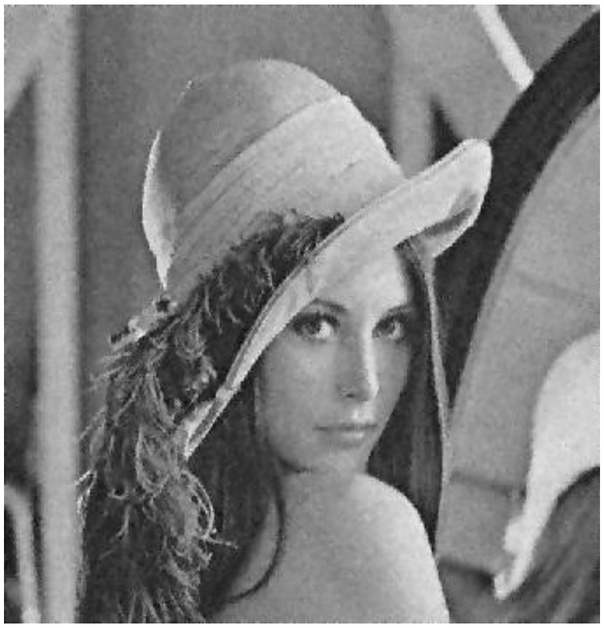


Figure 2.4: Wiener filtered image

A filter shall be applied on the image to smooth its appearance and decrease the noise level. The median filter is one of the most commonly used smoothing filters. It is used generally to reject any impulse type pixels of the image without touching its main features.

The process of median filtering is very simple and uses the median value from a group of pixels. The centre value of this group of pixels is then replaced by the median to ensure that no pulsed pixels exist. Following this procedure, every single pixel inside an image is processed individually and the median is used instead of it. In the case of noise, Median filter tends to detect that noise and replace it by the median of the surrounding elements. Figure 2.5 presents the application of median filter on a matrix of elements. In Figure 2.6, the application of Median filter on the noisy image of Figure 2.4 is illustrated. It is obvious that the Median filter has reduced the noise level.

$$\begin{array}{cccccccc}
24 & 23 & 21 & 20 & 18 & 17 & 16 & 17 \\
40 & 24 & 24 & 23 & 21 & 19 & 18 & 18 \\
21 & 23 & 35 & 24 & 22 & 20 & 18 & 20 \\
18 & 20 & 22 & 22 & 20 & 18 & 16 & 21 \\
17 & 18 & 19 & 18 & 17 & 16 & 15 & 20 \\
18 & 17 & 17 & 16 & 16 & 15 & 15 & 17 \\
19 & 18 & 16 & 15 & 15 & 15 & 16 & 14 \\
17 & 16 & 16 & 15 & 14 & 13 & 13 & 11
\end{array}
\xrightarrow{\text{Median Filter}}
\begin{array}{cccccccc}
24 & 24 & 21 & 20 & 18 & 17 & 17 & 17 \\
23 & 24 & 23 & 22 & 20 & 18 & 18 & 18 \\
21 & 23 & 23 & 22 & 21 & 19 & 18 & 19 \\
18 & 20 & 22 & 22 & 20 & 18 & 18 & 20 \\
18 & 18 & 18 & 18 & 17 & 16 & 16 & 17 \\
18 & 18 & 17 & 16 & 16 & 15 & 15 & 16 \\
18 & 17 & 16 & 16 & 15 & 15 & 15 & 14 \\
19 & 18 & 16 & 16 & 15 & 15 & 15 & 14
\end{array}$$

(a) Original (b) Filtered

Figure 2.5: Application of Median filter on an image matrix

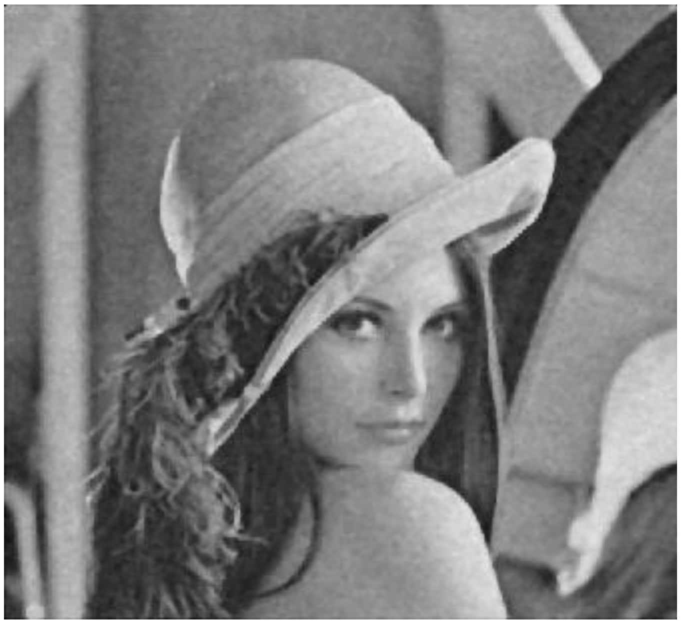


Figure 2.6: Application of median filter on noisy image

2.4 Image Segmentation

Image segmentation is one of the most important image processing techniques. It is defined as the process in which an image can be divided into groups of pixels that are interconnected between each other based on some criterions. The importance of image segmentation is due to the different fields where it can be applied. Segmentation tries to emphasize pixels that have common features and separate them together. The idea is to group the similar intensity pixels or the similar colour code pixels in groups and identify them. There are different methods of segmentation, however, they are not totally applied because they consume some time. Over more, there is no global segmentation method that is useful for any application. All segmentation methods that are based on intensity are very sensible to the case that different parts or objects have equal intensities. This will often cause wrong classification if the objects are not belonging to the same class from the point of view of the eye. Many segmentation techniques are being implemented and used for commercial and non commercial purposes. Image segmentation can be applied using three different categories, manual, automatic, and semiautomatic segmentation (Hodneland, 2003).

2.4.1 Manual segmentation

In this method, the pixels that belong to similar intensity groups can be manually extracted. However, the work on large images or huge number of images will be very time consuming and seems to be inefficient. The segmentation is done manually be drawing a contour around the similar pixels and identifying them.

2.4.2 Automatic segmentation

The process of automatically segmenting images is a very complex task that requires very high accuracy. The complexity of image segmentation automatically is due to the variation of images. Most of the segmentation algorithms require some knowledge about the images to be able to carry the segmentation process. With the lack of the priori information and the supervision the segmentation process may become arbitrary and can’t be guaranteed to outcome good results.

2.4.3 Semi automatic segmentation

Semi automatic segmentation is a combination of the two mentioned methods. It has the benefits of both methods by providing some initial parameters and offering some knowledge about the images. The automatic process is then responsible for the segmentation of the image. A semi segmentation process is applied to simplify the segmentation, reduce the time whenever the automatic segmentation can’t give the desired results. It is supervised by the user with the help of the computer or any processor type. It is faster and more efficient than the manual segmentation type.

2.4.4 Image thresholding

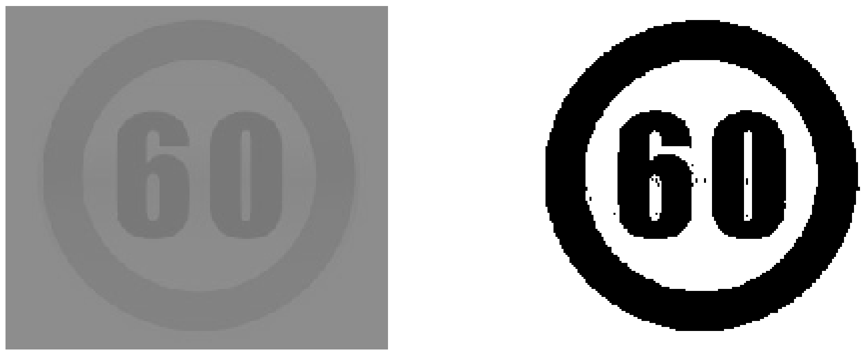
Thresholding is a very important image segmentation technique. It is a non linear operation that extracts a binary image out of gray scale image. The extraction is done based on a defined threshold value. If the pixel value is more than the threshold, the pixel value becomes 1, otherwise; it becomes 0. Different applications make use of thresholding for the separation of objects in the image into back ground and object. The object is the particular part that we intend to analyze. Thresholding provides a very easy and efficient way to separate foreground and background based on differences in colour intensity of the image. Supposing an image matrix defined by A, the thresholding process will give the output defined by (Santra, 2013) (Hodneland, 2003):

$$b(i,j)=\begin{cases}1, & a(i,j)\geq threshold\\ 0, & a(i,j)<threshold\end{cases} \tag{2.5}$$

After thresholding an image, the regions given binary value of 1 represent the object and the regions assigned with binary 0 represent the background of the image. Successful thresholding requires prior knowledge of the image properties. Generally, a best choice of the threshold is very important in thresholding. In most cases, the mean value is used for the thresholding purpose; however, the mean value is not always the best choice for threshold.

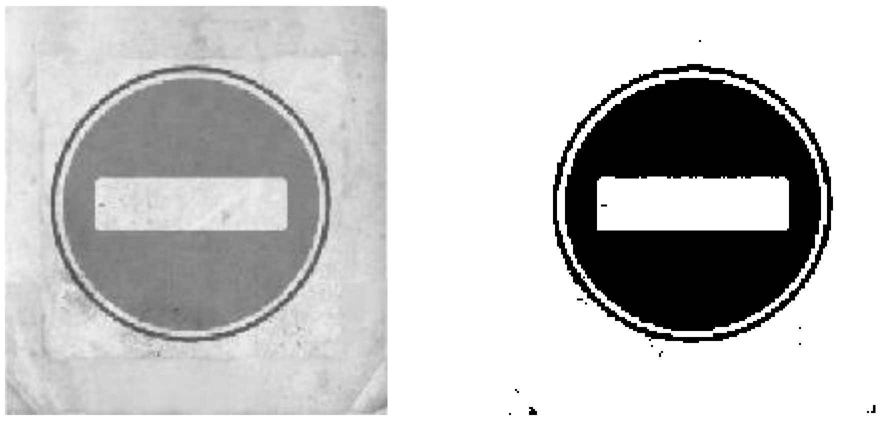
2.4.4.1 Selection of threshold value

The selection of threshold value is very important in the process of image segmentation. The simplest way use threshold segmentation is to use a fixed threshold value. This value can be the mean of the image pixels or any other value. Another way can be histogram based derived threshold. This threshold is extracted from the distribution of the image. Some algorithms use iterative methods in the segmentation of the images. An ISODATA algorithm is an example of the iterative threshold finding methods. The algorithm chooses an arbitrary threshold value and divides the image into two regions. The mean of the two regions is found and the threshold is updated based on it. The updating process continues until the threshold becomes constant (Hodneland, 2003).



(a) Original gray scale image (b) Segmented image

Figure 2.7: Image segmentation using mean threshold



(a) Original image (b) Segmented image

Figure 2.8: Image segmentation using fixed threshold (manual)

CHAPTER 3
ARTIFICIAL NEURAL NETWORKS

3.1 Overview

The Artificial Neural Networks (ANN) is computing devices or structures that are designed to imitate the human brain. They are generally assembled from tens or hundreds of smaller neurons or processing elements. These processing elements are connected between each other in a special architecture to guarantee the success of their function. Each one of these elements is a simple prototype of the real biological neuron. Biological neurons generate true outputs if they receive enough strong input signals. The output becomes null if the input signals are less strong than a given value. The connectivity strength between these nodes is variable and adjusted according to the task required from them (G. Anderson & McNeill, 1992). The idea of imitating the biological neural networks arisen early in the 1940s of the 20th century. Psychological scientists were fascinated by the way human brains worked. They tried to simulate their functions to create better understanding of the biological nervous system.

3.2 Principles of ANN

Flow of signals in biological neurons is considered one of the most complex processes that happen in the synapses of biological cells. Chemical substances are being released from the sender cell through synapses to the receiver cell. The receiver cell responds by creating an electric potential whose strength is a function of the chemical substance. This electric potential moves inside the body of the cell. The neuron becomes active if its potential is strong and reaches a threshold value. The active neurons create another signal and send it to the next neuron causing its activation. This simple idea is implemented in creating artificial models of the ANN structures. The artificial neurons presented in Figure 3.1 represent a simple illustration of the idea of using artificial neural networks. The inputs denoted (x_1, x_2, x_3…) are received by the neurons denoted (ω_1, ω_2,..). These signals are collected and weighted by the neurons. The neurons in turn submit all the collected inputs to a summing junction whose function is to decide whether to activate or not. In ANNs, the summing junction contains different functions that are applied on the received signals (Hebb, 1949).

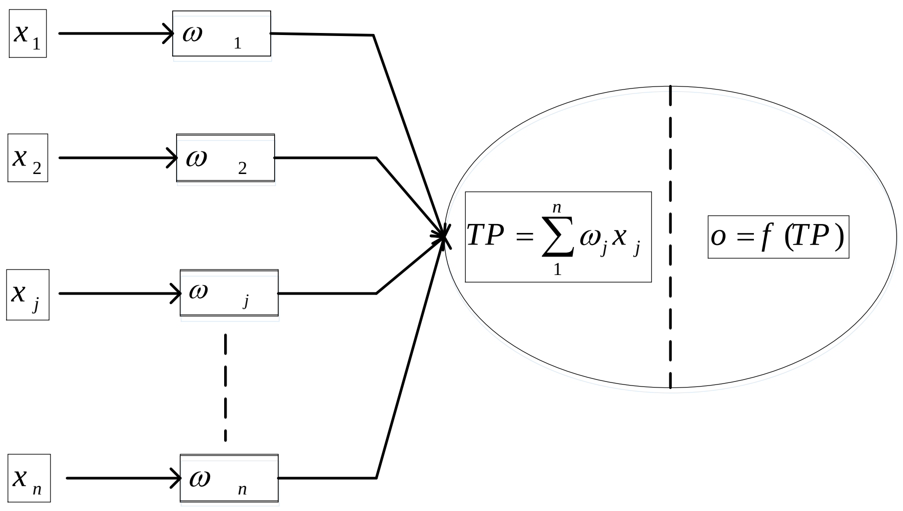


Figure 3.1: Simple structure of artificial neurons

The artificial neuron illustrated in the previous picture contains an arbitrary number of inputs. Each one of these inputs is connected to an adjustable weight. The weights are used to evaluate the inputs and adjust their strengths in the generation of the neural reaction. These weights are used to evaluate the output of the neuron. The formula of the output is described by:

$$TP = \sum \omega_i x_i \qquad (2.6)$$

The final output of the neuron is given as a mathematical function of the previous result. This function is called transfer function and described by:

$$o = f(TP) \qquad (2.7)$$

In the neural networks there are different types of transfer functions that will be discussed later in this chapter. The first type of transfer functions that has been used early was the threshold function. This function is also known by the term hard limit function. Nowadays, there are different types of transfer functions that can be used for the output of the network such as linear functions, sigmoid, and ramps (Seiffert, 2002). The system in an ANN structure is built of hundreds or thousands of neural interconnections and weights. These weights are distributed in multiple different size layers. The connections are varying according to the implemented topology of the network (Sekeroglu & Khashman, 2004). In the artificial neural networks, two types can be distinguished according to the learning paradigm of the network. These are the supervised learning and the unsupervised learning structures. In a supervised learning structure, the ANN uses well known sets of inputs and outputs to train the network. The unsupervised learning doesn’t need output examples to train the network (Zurada, 1992).

3.2.1 Supervised learning of ANN

In this type of ANN structure, the network receives its input and the correspondent output during the training phase. At each time the training process is activated, the generated results of the learning are compared to the desired outputs. Based on the error between the actual and the desired outputs a special algorithm is implemented to adjust the structure. The weights are being adjusted during the learning to ensure that the network is learning the examples (Caruana & Niculescu, 2006). One of the most famous learning algorithms of the supervised networks is the back propagation algorithm. Figure 3.2 presents the idea of the supervised learning paradigm in neural networks. The supervised learning is known also by the term learning by example (Roberts, 2015).

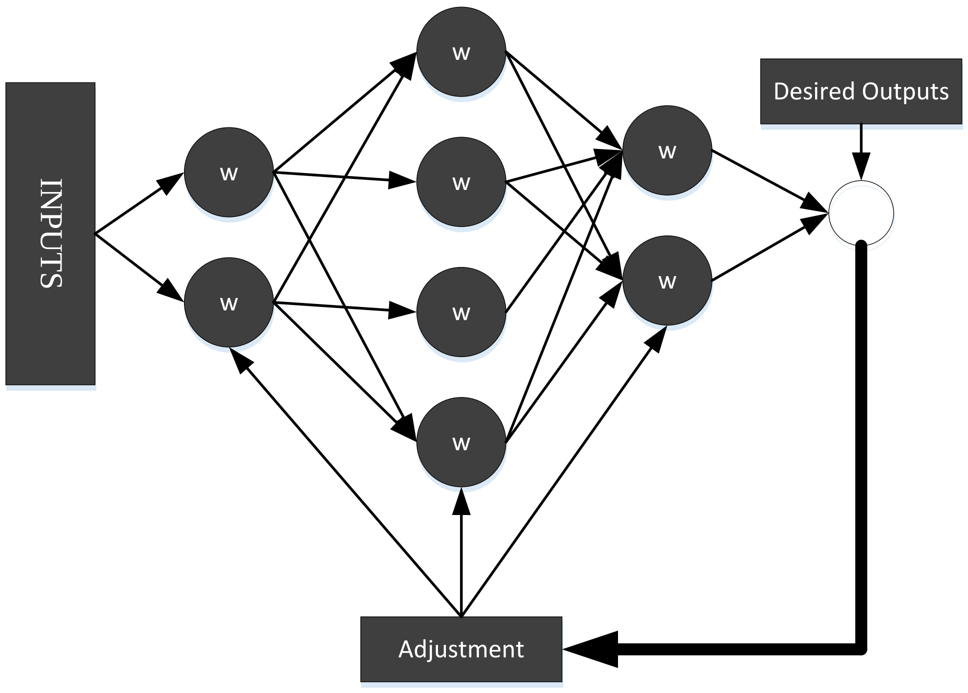


Figure 3.2: Supervised learning of the ANN

3.2.2 Unsupervised learning

The unsupervised learning of the ANN is a structure where the hidden layers of neurons are responsible to find a way to organize and adjust them without any external help. No examples or output samples are provided to the network in this type of network. The main types of the unsupervised learning neural network are Kohnen networks, and competitive learning networks. Figure 3.3 illustrates the structure of unsupervised learning network.

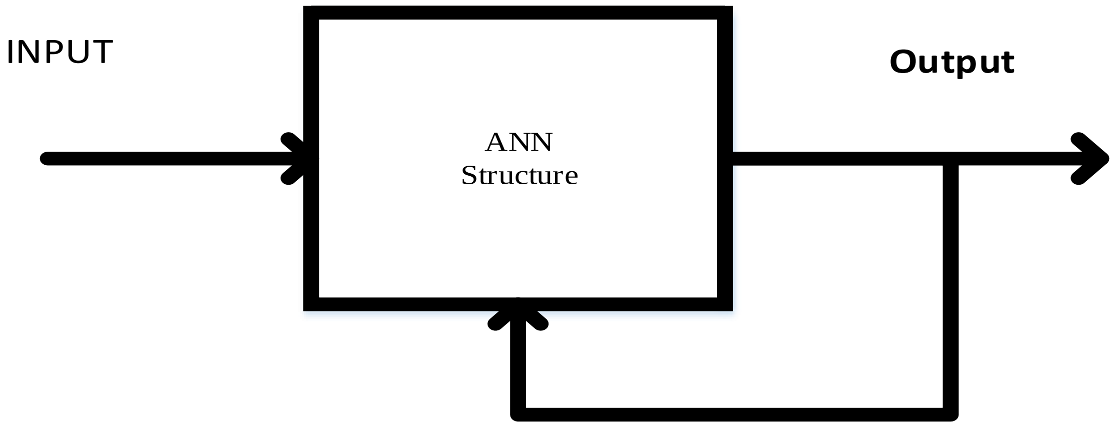


Figure 3.3: Unsupervised learning network structure

Competitive learning network is an example of this type of learning algorithms. The easiest method of implementing this learning algorithm is through connection of each weigh with all precedent weights. The main disadvantage of such algorithm is the lack of learning examples. Figure 3.4 presents a simplified structure of the competitive neural networks (Sekeroglu & Khashman, 2004).

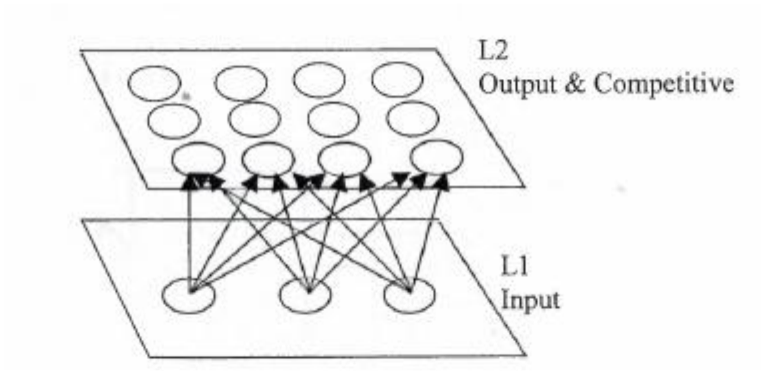


Figure 3.4: Competitive learning neural network

3.2.3 Activation functions in artificial neural networks

In the artificial neural networks, transfer functions are the functions used at the output of summation junction. They convert the total input signal of the summation junction into output within some limits. Transfer functions can be any type of derivable functions; they can be linear or non linear. The neural network is specified by the type of transfer function and its output is a function of the used transfer function. Linear activation functions like pure line function are used in neural networks. The function of this type of transfer function is defined by (Cios & Shields, 1997):

$$o = a * x \tag{2.8}$$

Where; “a” is a slope of the function that can be any real number. Below presents the input output curve of the pure line transfer function for different values of the slope.

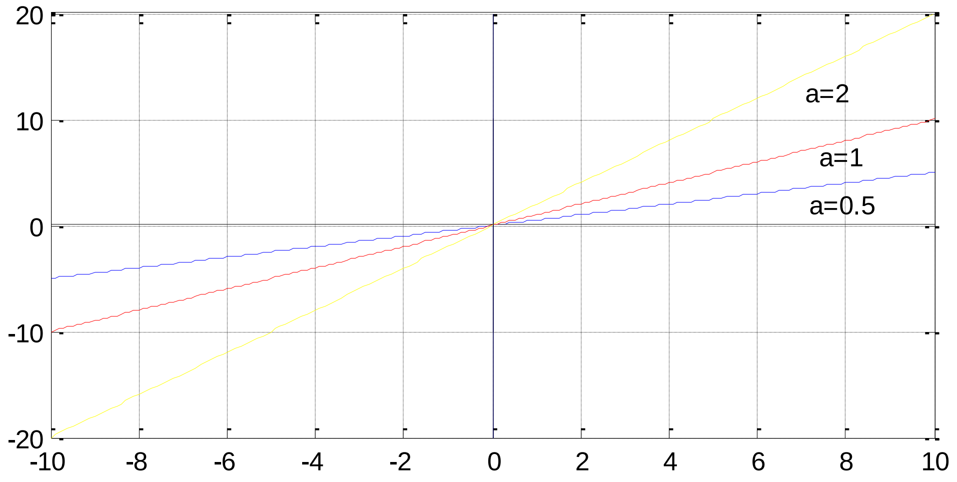


Figure 3.5: Pure line transfer function curves

Besides the pure line transfer function, two other types of transfer function are tangent sigmoid and logarithmic sigmoid. The output of these two transfer functions is given in Figure 3.6. It is important to mention here that these two functions offer the advantage that the output is limited within small hysteresis values. In a tangent function, the output is between -1 and 1 while in a logarithmic sigmoid, the output is between 0 and 1. Some types of transfer functions are linear in some zone and becomes non linear or saturated beyond that zone. Such functions are also useful in some ANN applications. It is still important to keep in mind that a function can be implemented in the ANN only if it is derivable in the period of its definition.

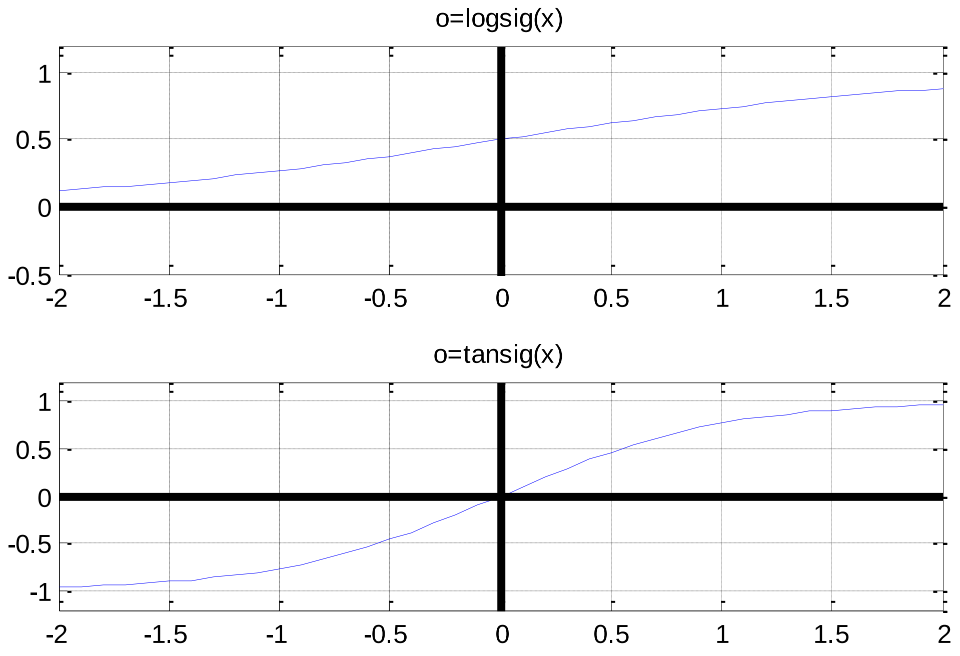


Figure 3.6: Tangent and logarithmic sigmoid functions

3.3 Back Propagation Learning Algorithm

The use of single layer perceptron network was limited by the capabilities of single layer network until the beginning of 80s of the last century (Sekeroglu & Khashman, 2004). The need for powerful artificial neural network structure motivated researchers to invest in the Windrow-Hoff learning rule to generalize it for multiple layered networks. The extension of this rule has led to the creation of the back propagation learning algorithm. It has been noticed that the proper teaching of the back propagation networks will generate logical and reasonable answers for questions and problems that were never seen.

Generally, a new input vector will generate similar output to those having common patterns with it. The ANN will be able to generate the correct result if it has been trained and learned the pattern from similar input output sets. This means that we can use different input output sets to teach the pattern in these sets and store it inside the network structure. The amazing function of the network is to be able to generalize these patterns for new sets that were never seen before. This leads to the main idea of training of the ANN in a supervised training process. Back propagation learning algorithm for ANN passes by three main stages.

In the first stage of the BPANN, the input set of patterns is forward fed to the structure of ANN. Calculations of weighted outputs and summations are performed in this process. The transfer functions are applied to generate final results. At the end of this process, the error signal is generated by comparing the outputs and desired outputs of the input sets. In the second step, the calculated error in implemented in the back propagation of corrective signals toward the layers' weights. The new weights are then found and used for a next feed forward process. These two processes continue looping until a stop criterion is met. It is sometimes useful to implement multiple hidden layers in some applications. The final step is the test process where a one feed forward process is applied to generate the outputs for new inputs.

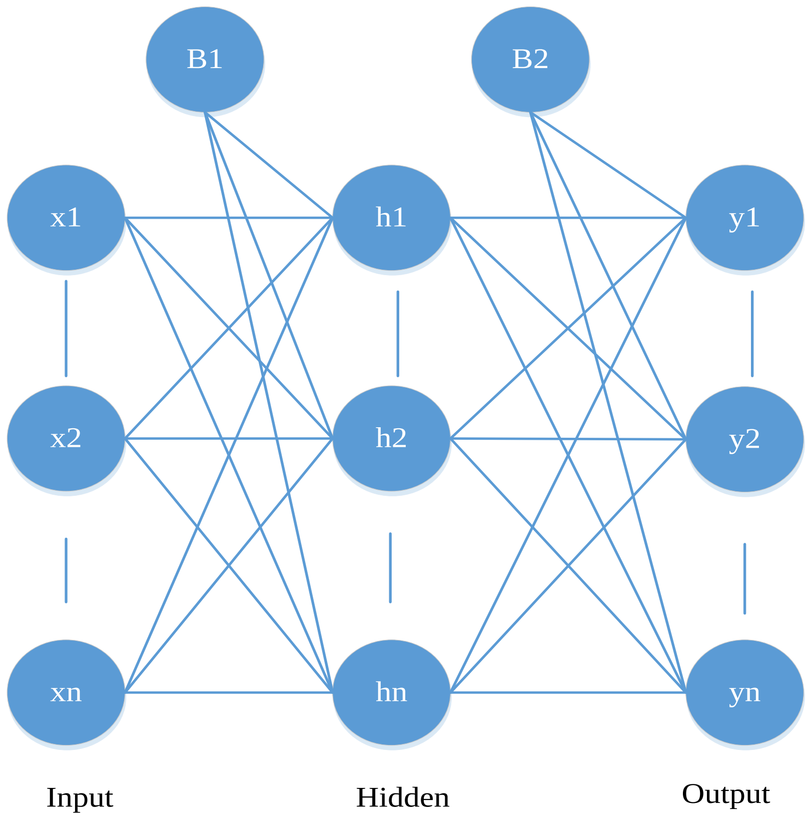


Figure 3.7: One hidden layer BPANN

3.3.1 Structure of BP algorithm

The diagram in Figure **3.7** presents a multilayer ANN structure that uses back propagation ANN. The structure consists of one input layer, one hidden layer, and one output layer.

Another extra layer is used to add some bias to the network outputs. The bias vector plays the rule of weights with the difference that their values are 1. In the figure, just the feed forward information flow direction is presented. The flow of back propagation signals goes from the output toward the input in successive order. The algorithm in this chapter will be discussed for one hidden layer. However, this algorithm is suitable for any number of hidden layers (Sekeroglu & Khashman, 2004).

3.3.2 Learning algorithm

The training process in a BPANN passes by three steps as discussed previously in this chapter. The feed forward of the inputs toward the output layer of the network is the first step in this process. The error back propagation toward the previous layers is the next step; while the last step includes the adjustment of the weights to reduce the error function. Each input neuron receives the input pulse and redistributes it toward the hidden weights. The hidden weights compute their outputs based on the activation function. Each hidden neuron submits its output to all output neurons where all inputs are also summed together. Another transfer function is applied on the output of each neuron producing the output of the network. At this stage, the feed forward step ends and the error calculation is going to be accomplished. The outputs are compared with the set of desired output and the error is found. Upon finding the error, an error signal is being propagated toward the output layer. The new weights of the output layer are then found. Another error signal is also sent to the hidden layers (Orr, Schraudolph, & Cummins, 1999).

When the propagation of the error reaches the input layer, a new feed forward iteration is processed to find the new output. This process continues until the error becomes minor. The flowchart of Figure **3.8** shows these steps in the form of block diagram.

3.3.3 Parameters of the BPANN

The success of BPANN algorithm is based on the choice of different parameters that can increase or decrease the efficiency of the algorithm. These parameters are the transfer or activation functions, the learning rate, and the momentum factor. In advanced BP algorithm structures, other parameters also can play an important role in increasing the convergence speed of the training (Bataineh, 2012).

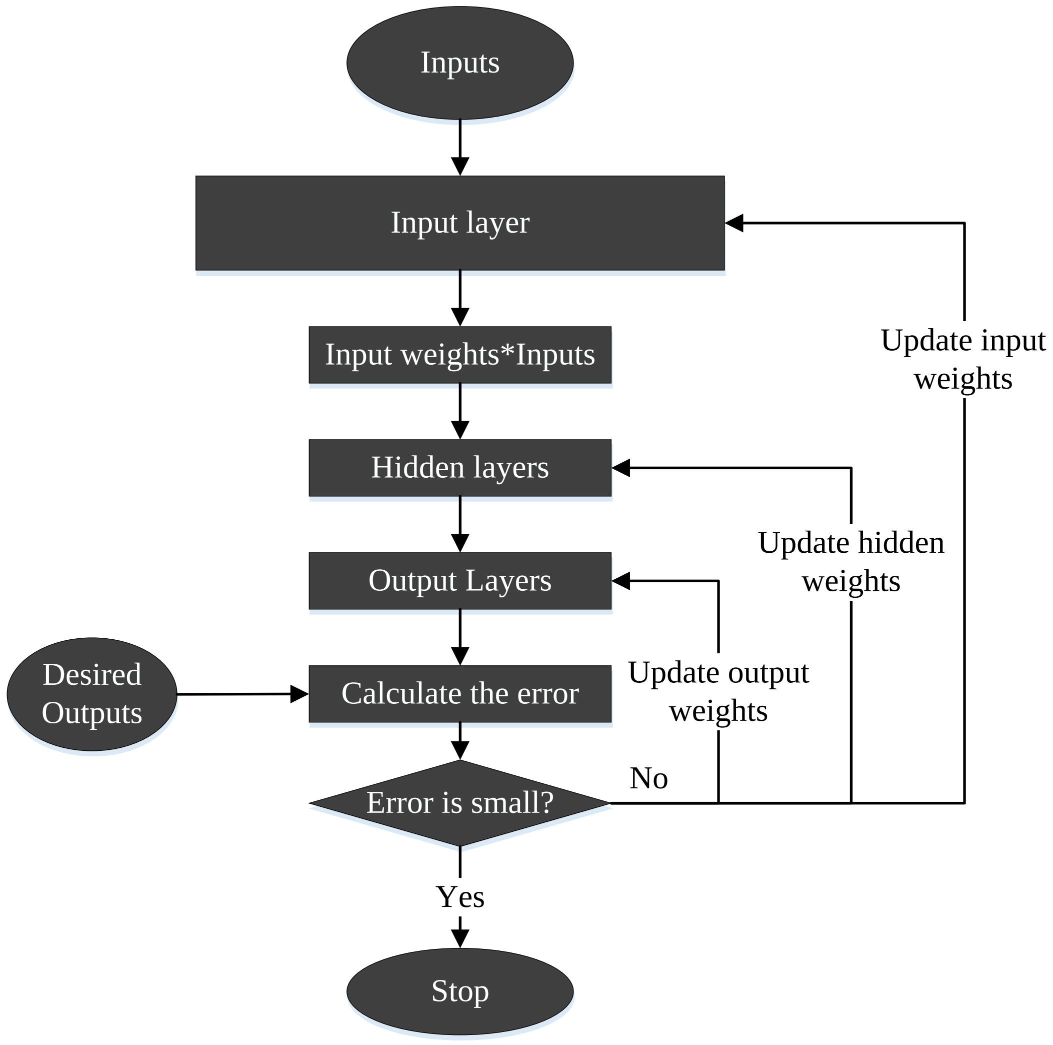


Figure 3.8: Flow chart of the BP algorithm

3.3.4 Learning rate

Learning rate is a scaling factor used to calculate the required variation in the weight during the training of ANN. In early neural network structures, the learning rate used to be fixed during the training. However, adaptation algorithms like the bold driver algorithm can be used to ameliorate the performance and search online for the best learning rate value. The variation in the weight that results from the feed forward iteration can be given by:

$$\Delta\omega_{jk}(t+1) = \lambda\delta_k h_j + \alpha(\omega_{jk}(t) - \omega_{jk}(t-1)) \qquad (2.9)$$

And the new value of the output weight is given by:

$$\omega_{jk}(t+1) = \omega_{jk}(t) + \Delta\omega_{jk}(t+1) \tag{2.10}$$

Where the term λ denotes the learning rate and the term α is called the momentum factor that will be discussed later (Orr et al., 1999).

3.3.1 Bold drive adjustment technique

This technique was proposed to ameliorate the performance of the ANN back propagation learning algorithm. Using this technique, the learning process of ANN can be enhanced and done faster than the normal training. The idea proposes that the learning rate is not fixed at a value but it will be adjusted each iteration. The learning rate is increased by a small amount and the error of the next iteration is compared with the actual iteration error. If the error is decreased, the learning rate is increased again; otherwise, it will be decreased by a small rate also. This process continues until the end of the training (Orr et al., 1999). The learning rate in this method is adaptive rather than being fixed and will search for the best value that guarantees its continuous learning. However, this idea has its own drawbacks because the higher learning rate can lead to the over fitting problem or memory of data rather than learning.

3.3.2 Momentum factor

The momentum factor is another term used in the adaptation of the weights of the network. It is useful to invest in the variations of the previous weights and include them in the new weight values. The momentum factor is very useful in avoiding the problem of local minima that happens during the training of the ANN. This problem happens when the program falls in a region where the error is less than the two neighbour points as shown in Figure **3.9**.b; the figure shows the value of the error which is less than the two surrounding points. This local minima problem will cause the program to continue oscillating around it without being able to go further during training. Momentum factor pulls out the error from that point and guarantees a solution of this problem.

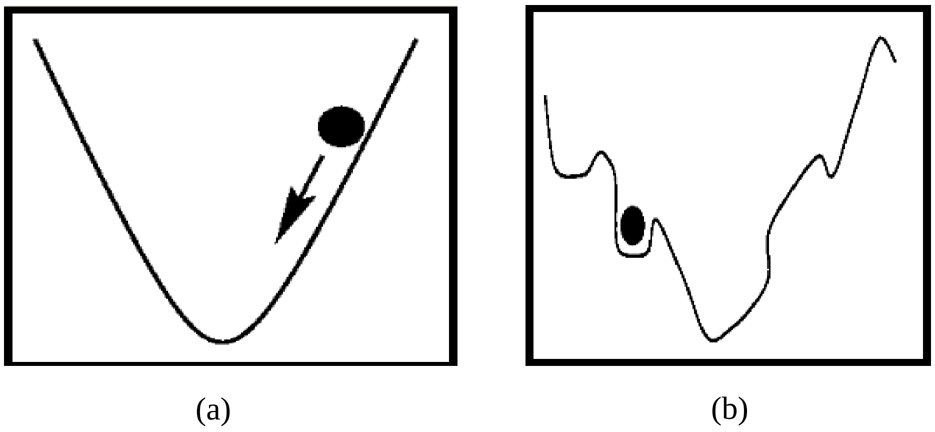

(a) (b)

Figure 3.9: The phenomena of local minima

3.4 Areas of Implementation of Artificial Neural Networks

Neural structures are nowadays implemented in many areas of science and for different types of applications in our life. They are widely implemented in thousands of researches around the world and in many scientific references. Neural networks are believed to be efficient in such a way that makes them able to do different task just like humans do. In some areas, the researches of ANN are in their basics while in other areas they have achieved great achievements. The technology of neural network is a future shaping technology that may opens the skies for unlimited development of different sciences. The discussion of the different areas of application of neural networks is endless and will take place in this part of our modest work. The neural networks can be found basically in pattern recognition applications, features estimation, approximation of functions, different types of control systems, smart antenna beam forming technologies, and memory (Hykin, 1999).

3.4.1 Pattern association

Pattern association is a memory that is similar to the brain learns by association of things. The neural network in this process will be expected to memorize a set of data after being presented to the network. This structure is known by the name auto association structure. It is a non supervised neural network process where no specific outputs are expected. Whenever the outputs are presented to the network in accordance with the input set, these inputs are associated with the chosen outputs in a supervised learning process. The process is then known by the name hetero association.

3.4.2 Feature recognition

Feature or pattern recognition is a very simple daily human process. This process needs no effort from the humans as they can easily recognize and classify things with the need of any mental efforts. Shapes, forms, things, and people are being classified by human brain every moment with no effort. The smells of foods and the taste of different things can be easily recognized by humans too simply. Persons can be recognized easily even if they do some changes to their appearance and whatever their ages have changed. Pattern recognition is defined as a practice that assigns a signal or input to a defined output pattern or to add it to a defined category (Hykin, 1999). Traditional processors find it very difficult to do classification or pattern recognition without the help of very intelligent algorithms. The ANNs are fantastic in doing the job just like humans if they were trained enough for it. A suitably learned neural network can effortlessly classify and identify features and add them to specific classes. Ear recognition, Signature recognition, palm recognition, and iris recognition are some of the examples of these applications (Bishop, 1995).

3.4.3 Function approximation

In numerical mathematics, interpolation and approximation of mathematical functions have very great importance. It is important to find the relation different types of variables. Different correlated groups of variables can be numerically interconnected through input output association functions. The neural networks are able to extract such types of relations associated between variables of different groups. There are mainly two different ways of describing or approximating functions using neural networks. The first is called system identification as shown in Figure 3.10 below. The modeling of the unknown system is achieved using the artificial neural network. In the training process, the unknown system and the neural network system are fed by the same inputs. The outputs of the two systems are then compared in continuous mode. The error between the two outputs is propagated back to the network such that it is being reduced. The process continues updating the network until the response of the two systems for any input will become similar. At that point, the neural network is considered as a model of the unknown system that has its features and can give its same response.

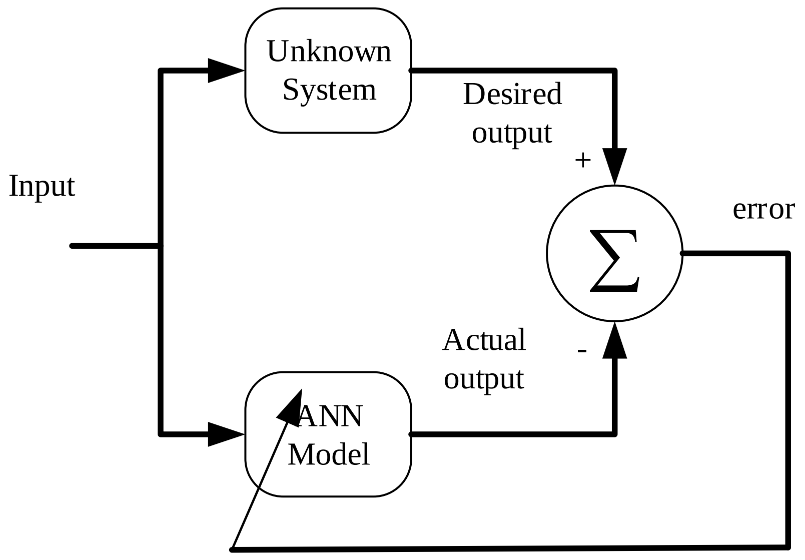


Figure 3.10: ANN for process identification

The second application is called inverse system identification in which a system is desired to be designed to cancel the output of another system at any moment. The inverse of the system in Figure 3.11 can be built based on the ANN just as the figure explains. The output of the system is used as input for the neural network system while its input is considered the desired output for it. This way, the neural structure will adapt its weights to produce the initial input whatever the effect of the system is. This is generally the case of filtering the communication systems signals. The system that is to be inversed is the noise collected from the transmission channels. The neural networks help in achieving the transmitted signal before the channel noise was added to it (D. Anderson & McNeill, 2010).

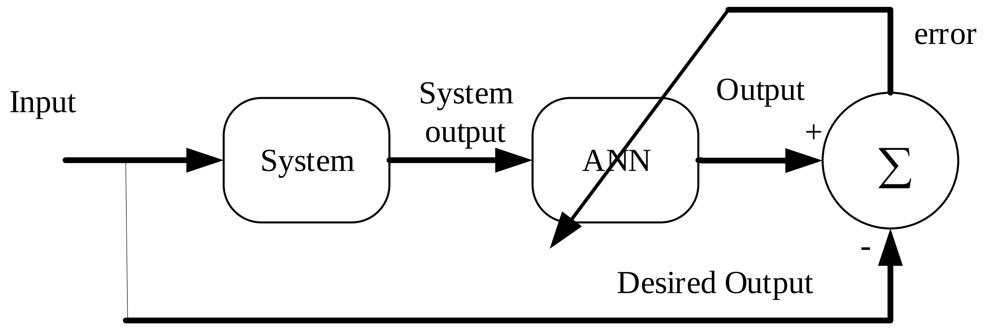


Figure 3.11: ANN used for inverse system finding

3.4.4 Control systems

The control system is a huge universe in which a lot of theories are being applied to achieve one task. Neural networks can be used in the domain of control systems to add new dimensions for the traditional control systems. The human brain is the main prove of capability of the artificial neural networks in the control of different processes in an intelligent way. Human's brain is a biological neural network that does control processes thousands of times in the second. It keeps the balance of the human all moments through measuring its position and controlling the muscles to keep in the correct position. It also helps the driver to keep tracking the road by using his eyes to see the position and then moving the steering toward the correct side of the road. Figure 3.12presents a simple control task of unity feedback to control the plan processes and generate the desired output all the time. The plant output is fed back to the controller to be compared with the desired output. The error signal is used to make adjustment of the weights of the controller such that the control signal drives the system in the right direction. A neural network based controller is used to achieve the required task easily. It forces the plant to follow its desired target with an intelligent adaptive way (Mehrotra, Mohan, & Ranka, 2001).

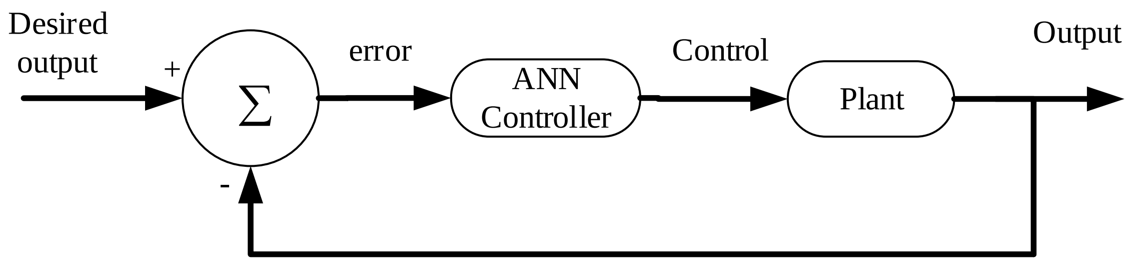


Figure 3.12: The ANN in the control systems

3.4.5 Filtering

Filtering is the operation in which special information is separated from mixed signals with noise. The noise contained in the signal is cancelled or rejected. Filtering process is important mission for different types of systems. Can't we imagine a microphone that can't filter our voice while we try to speak? Digital communication sets are mainly based on filtering to clarify our voice while speaking with somebody online or on loud speakers. Figure 3.13 gives a simple idea about the filtering task of different signals using artificial neural networks.

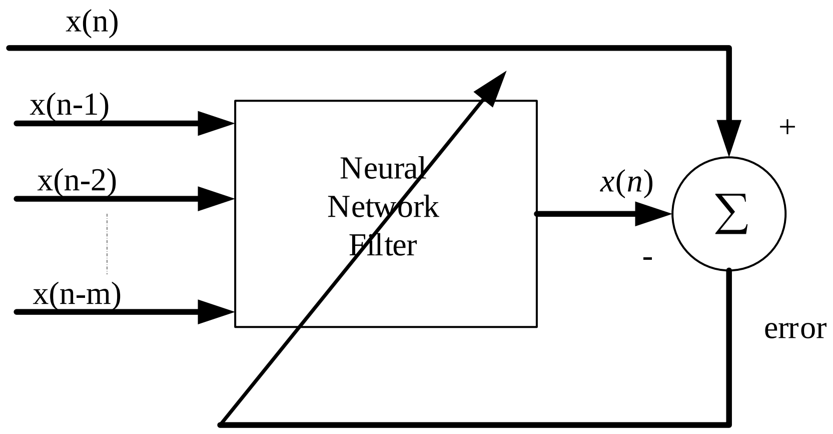


Figure 3.13: Artificial network for filtering

CHAPTER 4
RESULTS AND DISCUSSIONS

The chapter is the resume of the methods and practical experiments. The used methods and results will be presented and discussed in the course of this chapter. Different parameters of ANN will be examined and used. The implementation of the back propagation neural network for the training of traffic sign recognition system will be implemented in this chapter. Figure 4.1 presents a sample of the database implemented in this chapter. It shows 16 different images of a warning traffic sign. These images were created by adding different noises using MATLAB software, different types of noise, dust, and rain to the original image at the left upper corner of the figure. During this chapter, the training of a back propagation neural network based on different parameters will be discussed and presented. The use of the segmentation procedure shown earlier in this work will also be presented and discussed.

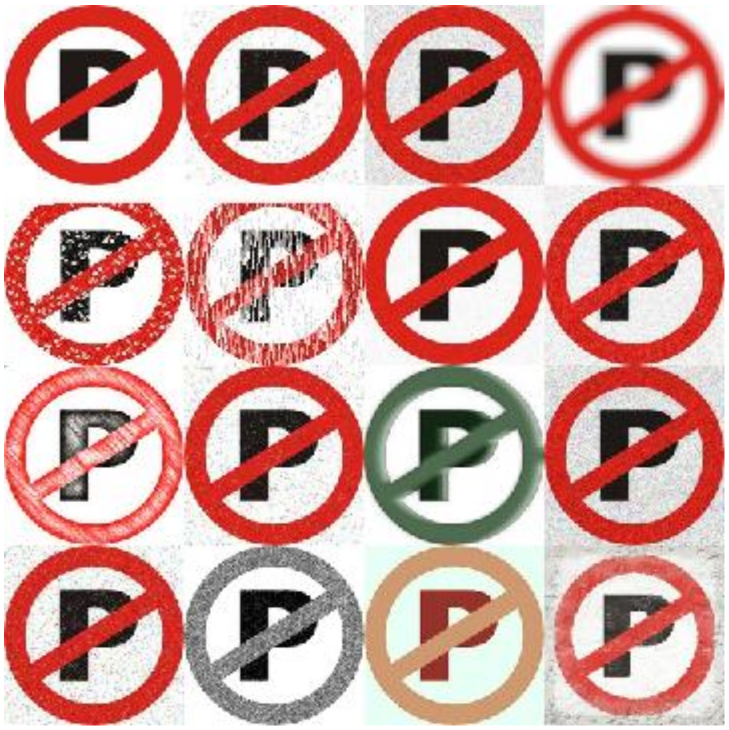


Figure 4.1: Sample of the image database of one warning sign

Data base images were created using MATLAB. Different noise types were added to the original traffic sign images to create variety of images and simulate environment effects like rain and sand on the traffic signs. Speckle noise, Salt & pepper noise, Gaussian noise, and Poisson noise were added to the original images. In some images, the red colour was replaced by another colour value to create new type of noisy images. In some images, arbitrarily chosen pixels were replaced by white colour to affect the clearance of original image. The original size of database images was 237*213 pixels.

4.1 Image processing

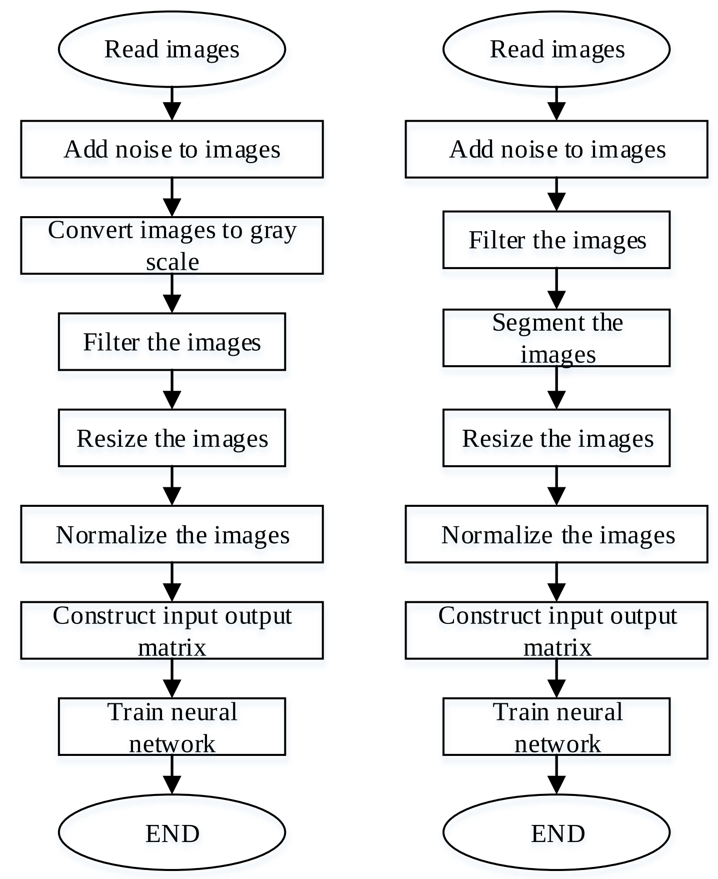
The database images were initially prepared for the implementation with artificial neural network by passing them through different image processing steps. The image processing applied on the database images was arranged as follow:

- Read the database coloured image.
- Convert the image into gray scale image.
- Apply Median filter on the image to remove noise.
- Normalize the image pixels.
- Resize the image using MATLAB resize instruction, size of 40*40 was chosen.
- Convert the resized image into vertical vector of length of 1600 elements.
- Arrange all converted images in an input matrix that is going to be fed to the ANN structure.

It is important to notice that the normalization of the inputs of the artificial neural network is a preferred process to simplify the learning of ANN. Normalization process can be ignored in our case as the image pixels have defined range. Normalization is very important with data that has different range or distribution characteristics to avoid miss interpretation by the ANN.

The Figure 4.2 presents the steps of image processing represented in the form of bloc diagram.

Figure 4.2-a show the steps of processing without applying the segmentation or edge detection on the traffic signs images. Figure 4.2-b present the steps applied with image segmentation. It is worth to mention that the implemented image segmentation method doesn’t need conversion of RGB images to gray scale. The image filtering was applied on the individual matrices of R, G and B components of the image in this case.



(a) Without segmentation (b) With segmentation

Figure 4.2: Bloc diagram of the image processing steps

4.2 Output coding of the images

In order to simplify the translation of network results and training of the system; a simple output coding of the targets is required. Each one of the signs has to be assigned suitable output code that will be generated once it is fed to the network. During the training, the output codes are assigned to the correspondent input images so that the neural network can learn the image pattern. As the total number of studied signs was equal to 13 signs, an output code for each one of them was simply assigned as shown in Table 4.1 below. Binary numbers are of the most simple and easy to use coding types. They were used here because the output can either be 0 or 1. This is suitable for the sigmoid transfer function types. Each column in the table below corresponds to one of the 13 traffic signs of our work.

Table 4.1: Output (target) coding of the 13 different traffic signs

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1

4.3 Experiments

Different experiments with different ANN structures and parameters were used in this work to validate the proposed methods. The final topology of the neural network used in this work is presented in Figure 4.3.

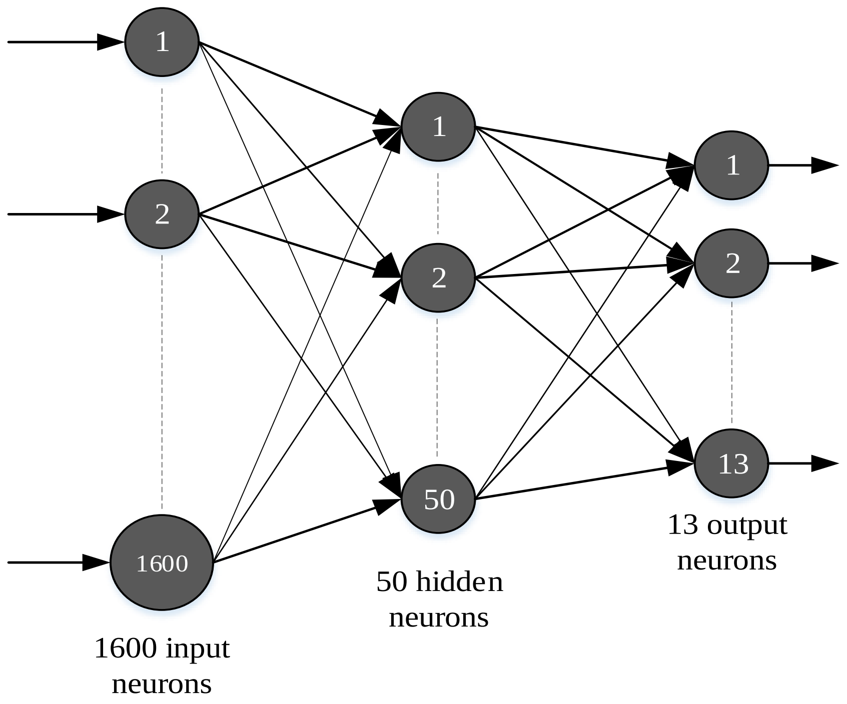


Figure 4.3: Final optimum topology of the implemented ANN

4.3.1 Experiment 1

In the first attempt, a simple one hidden layer based neural network is going to be used. The parameters of the network are given in Table 4.2. The results obtained in this experiment have shown good performance of the network without the need for any extra processing of the images. The training of this structure was very fast and has finished in 8 seconds. 280 training epochs were processed till the final training MSE became suitable and less than 0.000015. 119 images of the training group were all recognized with no errors. The training performance was 100%. Among the 89 test images group, 5 images were either not recognized or uncertain. The test efficiency of this group was 94.38%. The total system efficiency was approximately 98.56% with 203 correctly recognized images out of 208 total images. Figure 4.4 and Figure 4.5 illustrates the development of the error function curves of the training and test data respectively. It is noticed that the error curve was decreasing perfectly for both groups and reached very small values at the end of training. The final test MSE value was approximately 0,005. The training and test outputs of neural networks as response of one image for each sign are presented in Table 1 and Table 2 in the appendix 2. The tables show that the outputs converge perfectly to the exact output with very minor errors that don't affect the overall evaluation of the results. The individual error of each value was computed to be 0.01 for all training and test results.

Table 4.2: Parameters of the first used network structure

Training images	119	Training	Back propagation	MSE	$1.44*10^{-5}$
Test images	89	Transfer functions	"logsig"	H. neurons	50
Total images	208	Training time	8 seconds	Out. neurons	13
Hidden layers	1	epochs	280	In. neurons	1600
LR	0.02	MF	0.2	thr	0.4
Train perf.	100%	Test perf.	94.38%	Total perf.	98.56%

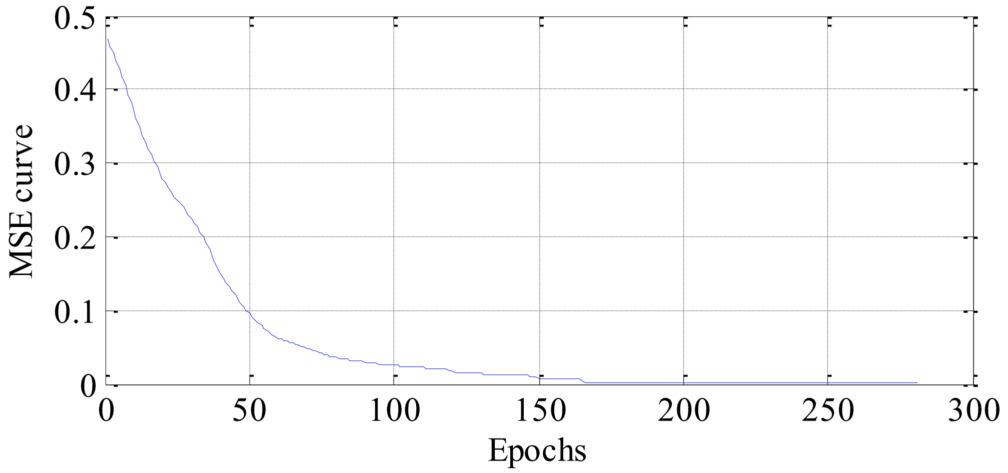


Figure 4.4: MSE performance curve of the first structure

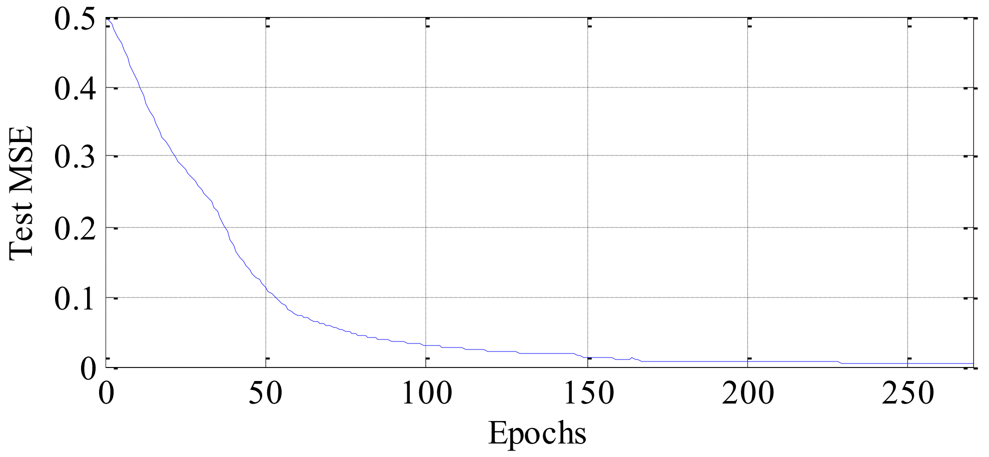
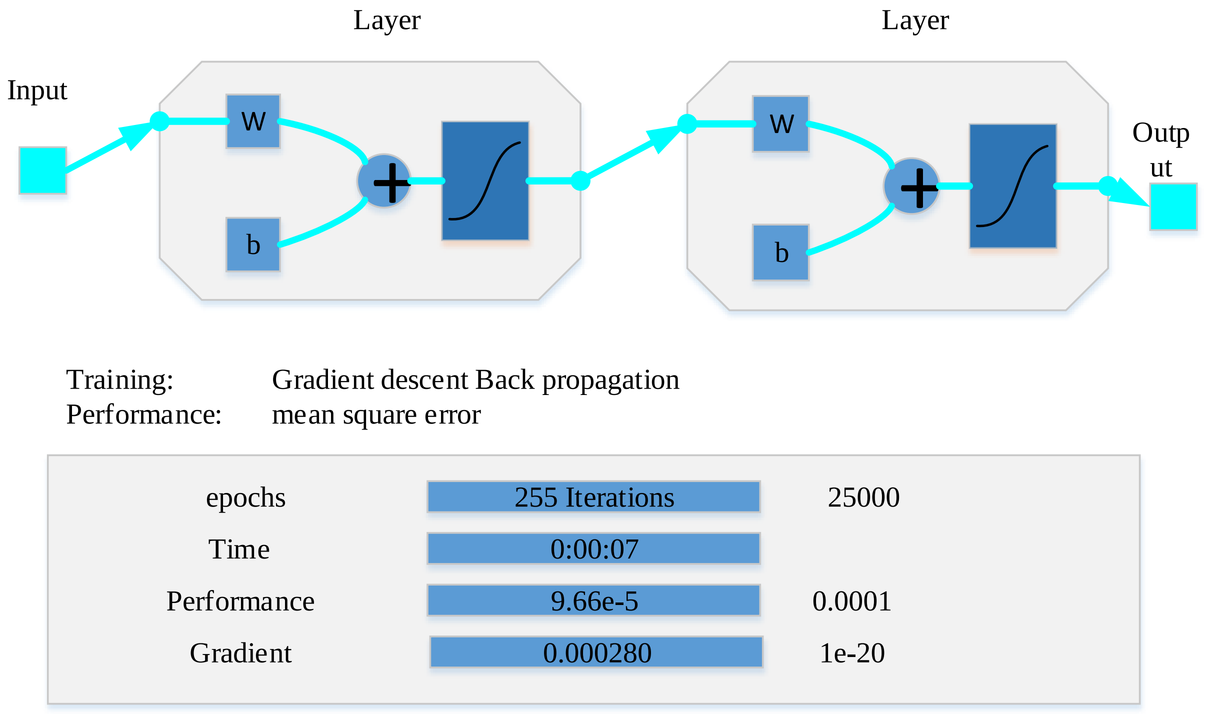


Figure 4.5: MSE curve of the test data during the training

4.3.2 Experiment 2

In this part, another combination of image numbers will be used and examined. Less images are going to be used for the training of the network. The database will be divided into two equal parts, one for training of the network and the other for the test of network. 8 images of each sign will be used for training while the other 8 images are going to be used for the test of the structure performance. The test images are not going to be taught to the network during the training process. All the other parameters are going to be kept unchanged. Figure 4.6 shows the artificial neural network training tool of MATLAB during the training of the network. It is seen that the training of this system has finished in 7 seconds and achieved an MSE of 0.0000966 after 255 iterations. After analyzing the results of training and test it was found that all the 104 training images were recognized correctly. Among the 104 test images, 100 images were recognized correctly and 4 images were badly identified. The training and test performance were found to be 100% and 96.15% respectively. The total performance of the system is 98.08%.

Table 3 in appendix 2 shows the output obtained from the network after feeding the network with 10 arbitrary test images. The table shows that the maximum individual error was less than 0.2 in the worst case.



Training: Gradient descent Back propagation
Performance: mean square error

epochs	255 Iterations	25000
Time	0:00:07	
Performance	9.66e-5	0.0001
Gradient	0.000280	1e-20

Figure 4.6: ANN training tool during the training of system

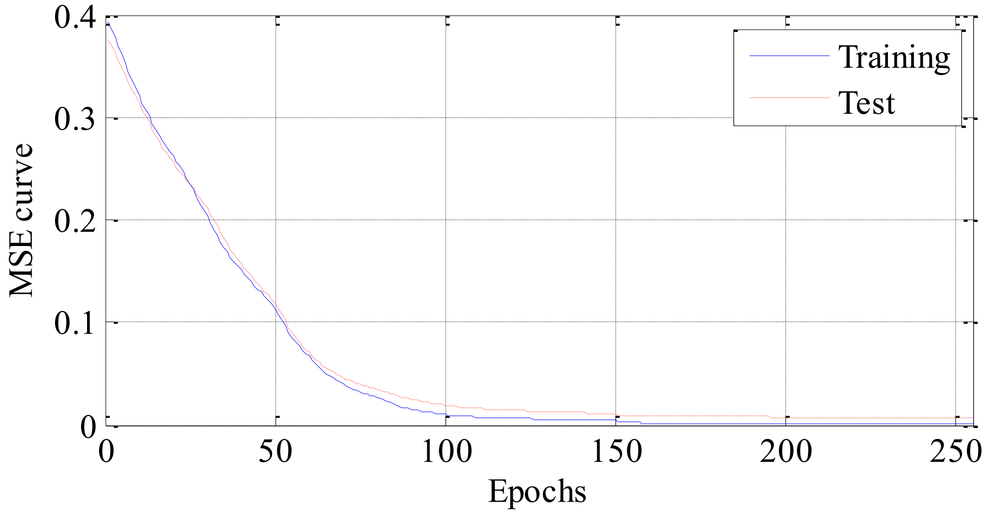


Figure 4.7: MSE curves of test and training images recognition

4.3.3 Experiment 3

In this experiment 4 training images out of the 16 images of each sign were used in training process. The same parameters as the previous section were used to test the program performance. The training time in this experiment was about 8 seconds and the total iteration number was 299 iterations. The training MSE was reduced to 0.00045 while the test MSE was found to be 0.0024. Table 4 of appendix 2 shows the training results of the system using 10 different input images of 10 different signs. It was noticed from the results that the individual error was increased in this experiment. As it is seen in sign S13, the value of the output was 0.56 with an error of 0.44 from the correct value. This result was accepted as the error is less than half the distance between the two possible values (0, 1).

After evaluating the results, it was found that the training images were all recognized correctly with 100% performance. Out of 156 test images, 9 images were not recognized. This has reduced the test performance to 94.23%. The total performance also was reduced in this experiment to 95.6%. The curve shown in appendix Table 4 illustrates the change in the training and test MSE curves during the training of the network.

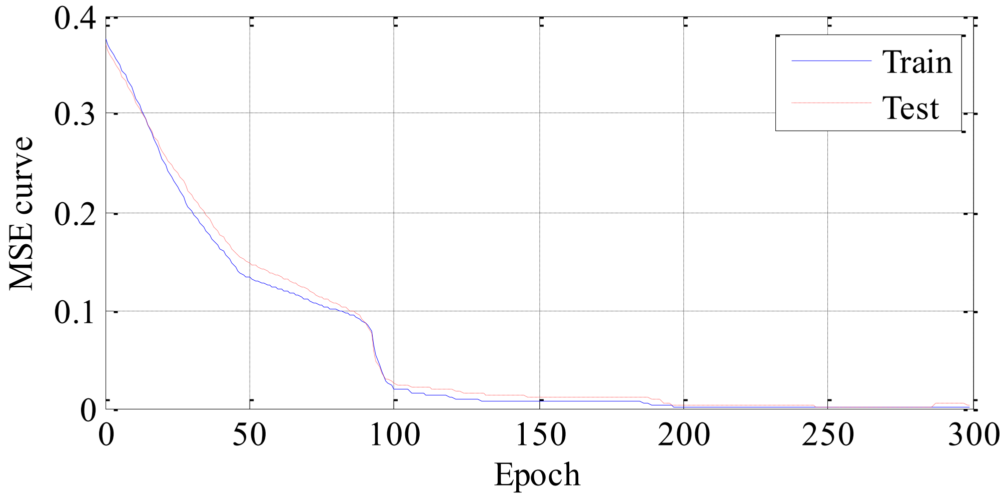


Figure 4.8: Training and test MSE curves

4.4 Experiment with Different Parameters

In order to examine the validity of the neural networks under different parameters of the training, many experiments were carried out with different parameters. At each experiment, the values of learning rate, momentum factor, MSE, iterations, and layers sizes were varied. The results of all these experiments were collected and tabulated.

Table 4.3: Comparison of different parameters and efficiencies (No segmentation)

	Train image	**Test image**	**LR**	**MF**	**epoch**	**Hid. Neu.**	**Train Eff.%**	**Test Eff.%**	**Total eff.%**	**MSE**	**T (s)**
Conf. 1	52	156	0.008	0.2	25000	5	84.6	61.5	67%	0.017	550
Conf. 2	52	156	0.008	0.2	468	20	100	84.62	88.5	0.0000	8
Conf. 3	52	156	0.008	0.04	379	20	100	90.36	92.79	0.0001	7
Conf. 4	83	119	0.008	0.04	258	40	100	95.2	97.12	0.0001	6
Conf. 5	83	119	0.8	0.9	1284	60	100	88.8	93.27	0.0001	38
Conf. 6	73	129	0.01	0.09	240	50	100	83.7	89.4	0.0001	6

The Table 4.3 presents the results of applying neural network with different training parameters on the system. It shows that the efficiency of the system is affected by the different applied parameters. The table shows that the use of smaller values of the learning rate and momentum factor increased the efficiency of the ANN. However, this assumption is not guaranteed to be true always. There are times where the use of high learning rate forces the network to memorize the data rather than lean the pattern of data. Memorizing the data is not a good practice of the network. This leads us to mention here the problem of over fitting in the neural networks where the network memorizes whatever it was trained to recognize without learning the features out of it. In the case of over fitting, the network can recognize the training images easily but will fail to recognize correctly the test mages. This phenomenon is well known phenomena in the artificial neural networks that need to be taken into account while designing a network.

4.5 Image Segmentation

In this part of the work, the traffic sign images are going to be segmented to extract the shape of the sign from the image before applying the artificial neural network. The segmentation is a very well known image processing technique that is widely used for extraction of important features from images. In general, it is done manually as there is no specific method for the automatic image segmentation that guarantees best results. In this work an algorithm is implemented to extract the image features before implementing the artificial neural networks. The algorithm that was explained in previously in section 2.10 of chapter 2 has given good results as will be discussed in this part of the work. This algorithm has shown good segmentation results of traffic signs. This algorithm was implemented by (Ellahyani et al., 2016) to overcome problems related to illumination changes and deterioration of colour. The segmentation process is very simple and based on an approach to convert an RGB image to a single intensity image "f" using the next relation:

$$f\,(R,G,B) = \frac{|R-G|+|G-B|+|B-R|}{3D} \qquad (4.1)$$

As the relation shows, the new matrix f is extracted directly from the RGB image without the need for conversion to gray scale. The value of D is adjusted based on the obtained results in order to achieve the best segmentation results. The value of D=20 was chosen and found to be satisfactory for our work. After finding the intensity matrix "f", the segmented image is easily found by applying a thresholding process on the matrix f. the thresholding process converts the matrix f into a binary image in which the region of interest appears clearly on a black back ground. The thresholding is applied using the next equation:

$$I(i,j) = \begin{cases} 0 \; ; \; f(i,j) < 1 \\ 1 \; ; \; f(i,j) \geq 1 \end{cases} \tag{4.2}$$

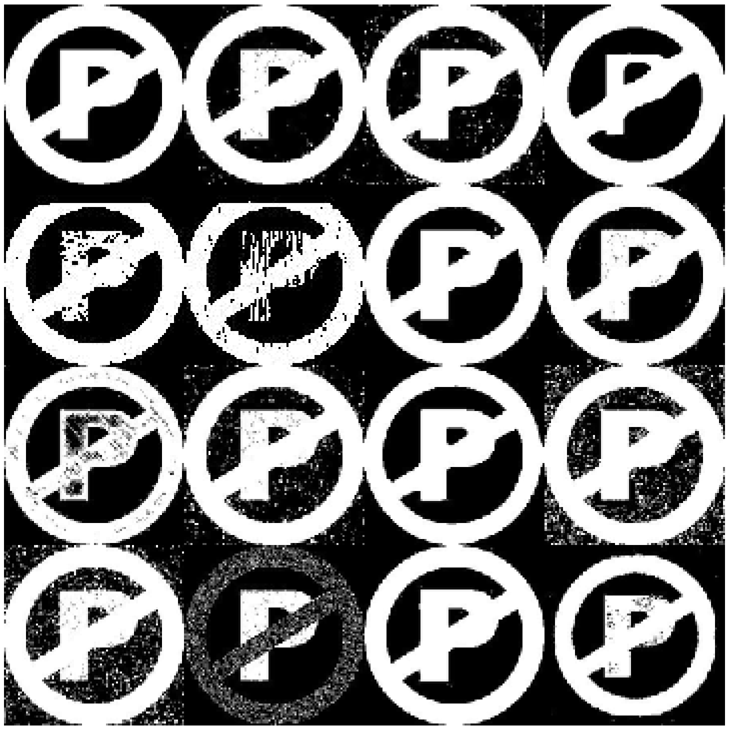


Figure 4.9: Sample of segmented images from RGB images

Where; ‘I’ is the image that results from the segmentation process. It was found that this segmentation method offers good segmentation efficiency and overcome the illumination variations during the day on the traffic signs. Figure 4.9 shows a sample of the segmented images of one warning sign. The figure shows that the segmentation method was successful in extracting the main features of the image. The images of the Figure 4.7 were firstly converted from RGB to gray scale and then segmented using the method presented in section 2.10. Some of the images were perfectly segmented while others were partially segmented due to the noise in the image. The training of the first set of images was carried out using 31 images for the training of the network. The parameters of the network during this experiment are given in

Table 4.4. Different experiments were applied on the set of images and the results were printed in the table.

The output of the neural network of the first experiment C.1 is presented in Table 5 of appendix 2. This table shows how accurate were the results of the training of the network in finding the output. Most of the outputs were generated with minimum MSE error. Test results of the first configuration are printed in the Table 6 in the appendix 2. The table shows that output was very similar to the desired or expected output for most of the images.

Table 4.4: Parameters of the ANN applied on the segmented images

	Train	**Test image**	**LR**	**MF**	**epoch**	**Hid. Neu.**	**Train Eff.%**	**Test Eff.%**	**Total eff.%**	**MSE**	**T (s)**
C. 1	32	177	0.01	0.09	218	30	100	82.5	85.1	0.00009	4
C. 2	32	177	0.01	0.09	704	100,10	100	75.7	79	0.0001	31
C. 3	32	177	0.01	0.09	180	100,40	100	80	83	0.0001	8
C. 4	52	156	0.01	0.09	180	100,40	100	89.7	92.3	0.0001	8
C. 5	52	156	0.01	0.09	164	80	100	83.3	87.5	0.0001	6
C. 6	104	104	0.01	0.09	208	50,50	100	84.6	92.3	0.0001	6

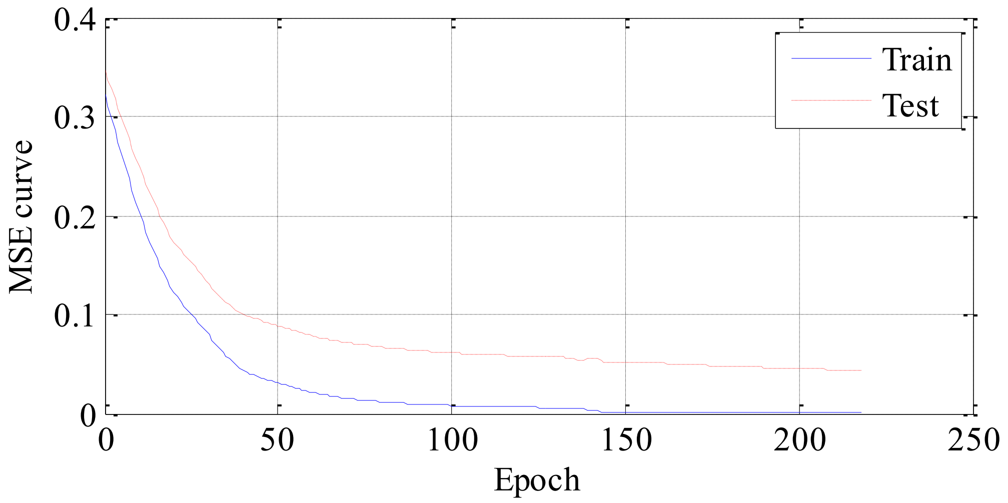


Figure 4.10: Error curves of the training results (C.1)Table 4.5 below presents the different results obtained in the two different cases discussed earlier. It is obvious that the use of segmentation has reduced slightly the efficiency of the artificial network in some cases. In most of the cases, the segmentation process has affected the test results although the training

results were not affected. It can be explained by the fact that the segmentation keeps less feature to be used for the training and test of the NN that that of the original image.

Table 4.5: Comparison of the obtained results with and without segmentation

	Train	Test image	LR	MF	Hid. Neu.	Train Eff.%	Test Eff.%	Total eff.%	MSE	T (s)
With segmentation	32	177	0.01	0.09	30	100	82.5	85.1	0.00009	4
	32	177	0.01	0.09	100,10	100	75.7	79	0.0001	31
	32	177	0.01	0.09	100,40	100	80	83	0.0001	8
	52	156	0.01	0.09	100,40	100	89.7	92.3	0.0001	8
	52	156	0.01	0.09	80	100	83.3	87.5	0.0001	6
Without segmentation	52	156	0.008	0.2	5	84.6	61.5	67%	0.017	550
	52	156	0.008	0.2	20	100	84.62	88.5	0.0000	8
	52	156	0.008	0.04	20	100	90.36	92.79	0.0001	7
	83	119	0.008	0.04	40	100	95.2	97.12	0.0001	6
	83	119	0.8	0.9	60	100	88.8	93.27	0.0001	38
	73	129	0.01	0.09	50	100	83.7	89.4	0.0001	6

4.9 Comparison of Results with Previous Works

In (Kobayashi et al., 2015), the authors introduced the use of genetic algorithms for the traffic sign recognition purpose. The overall recognition accuracy of their proposed system was claimed to be 94.9%. The author in (Kale, 2015) have presented a model based on the use of statistical analysis and neural network system using MATLAB software for the recognition of traffic signs without presenting any measure of success for his research. (Yihui Wu, Liu, Li, Liu, & Hu, 2016) has presented the work on traffic sign recognition using simple convolution neural network and cascade convolution neural network. They have shown that the simple CNN gave an overall performance of 90% while for the cascade CNN a performance of 98% was achieved for different types of traffic signs. The recognition of triangular and circular traffic signs in difficult images was proposed in (Zhe, Ren, & Bao, 2016). They presented a recognition method for the traffic signs that can extract the external form of the sign with good accuracy of 84%. (Bin, Ran, Yao, & Naiqiang, 2016) have also presented their work on the traffic sign recognition using SVM and ANN structure. They use 1100 traffic signs from the Chinese traffic dataset. No exact results were provided in their research although it was claimed that it presented better results. (Youfu Wu & Zusheng, 2016) have presented their work discussion about the segmentation of traffic sign images based on three different algorithms. The efficiency of their algorithms was reported to be 93, 91.4, and 96%. Their algorithm was mainly used to extract the traffic sign from the image without using sign recognition methods. An extraction method of the traffic sign from a video of 4 minutes length was presented in (Bengaluru, 2016). The method used ANN in the recognition of the signs with reported accuracy of 93.3%. In our work, we are implementing the artificial neural networks for the recognition of traffic signs. 13 different signs were used in our experiment. 16 different images for each sign were used with different types of noise. The dataset was divided into two separate parts; one for training and one for test purpose. The two parts were equal and contains 8 images for each sign. Training performance was 100% while test performance was 89.4%. The overall performance of the system was 96.15%.

CHAPTER 5
CONCLUSIONS AND FUTURE WORKS

In this work, a study of the implementation of artificial neural networks for traffic sign recognition. The studied system used 13 different traffic signs with 16 different images for each sign to apply training and test of the neural network. In order to consolidate and improve the structure of artificial neural networks, the system was assisted with some image segmentation process. The segmentation of the captured traffic sign images helps emphasizing on some required features while ignoring other undesired features prior to the application of artificial neural network. This process increases the performance of the artificial neural networks and helps obtaining better results out of the system.

The study of traffic sign recognition system is very important due to the modern developments in the field of smart systems that include smart homes, smart phones, smart cars, smart farms and many other smart technologies. These technologies investigate in the capability of computer devices and processors to built systems that can autonomously behave and condition with the changes that happen in the surrounding environment. Smart homes for example have the capability to keep track of sun shines, time, illumination, temperature, power consumption and behave accordingly to offer comfort environment without the need of any human intervention. Smart cars are able to self drive and park based on the smart systems provided. These cars drive in the roads safely with the least danger and error. The study of traffic sign recognition is important in assisting the function of smart cars by providing suitable idea of the road and required warnings. It is also important to help drivers to keep aware of the different warnings and instructions about the roads.

The proposed system implements the back propagation training algorithm for the training of multi-layer neural network with multiple hidden layer structure. The training was carried out using MATLAB software. Different training parameters were implemented in the training of the proposed system to explore the best possible performance parameters.

This work illustrates that the artificial neural networks can offer high performance in the recognition of traffic signs. The application of the artificial neural network has given results that ranged from 67% performance to 97.12% for different configurations. Some configurations have given an overall performance of 93% while other had a performance of 88%. From the obtained results in this work, it was found that the segmentation of images had an adverse effect on the performance of testing the neural network. It shows that the best overall performance obtained was 92.3%. It was noticed that the segmentation of traffic signs has reduced the test performance of the system.

Comparison of our work with previous similar works has shown that our results were superior to many other previous works and within the range of some works else. It is important to mention that the comparison doesn’t take in consideration the amount of database although some previous database was huge. These results prove that the artificial neural networks are capable of performing traffic sign recognition with high performance and reliability.

As future plans, this work can be extended to find new algorithms for traffic signs real life capturing, extraction and recognition.

REFERENCES

Anderson, D., & McNeill, G. (2010). *Artificial Neural Networks Technology A DACS State-of-the-Art Report*. NewYork.

Anderson, G., & McNeill, D. (1992). *Artificial Neural Networks Technology*.

Bataineh, M. (2012). *Artificial neural network for studying human performance*. University of Iowa.

Bengaluru, T. (2016). An Approach Towards Efficient Detection and Recognition of Traffic Signs in Videos using Neural Networks. In *IEEE WiSPNET2016* (pp. 456–459).

Bin, T., Ran, C., Yao, Y., & Naiqiang, L. (2016). Robust Traffic Sign Detection in Complex Road Environments (pp. 101–105).

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press.

Caruana, R., & Niculescu, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. In *International Conference on Machine Learning* (pp. 161–168). https://doi.org/10.1145/1143844.1143865

Cios, K. J., & Shields, M. E. (1997). The handbook of brain theory and neural networks. *Neurocomputing*, *16*(3). https://doi.org/10.1016/S0925-2312(97)00036-2

Ellahyani, A., Ansari, M. El, & Jaafari, I. El. (2016). Traffic sign detection and recognition based on random forests. *Elsevier*, *46*, 805–845.

Gonzalez, R. C., & Woods, R. E. (2001). *Digital Image Processing* (2nd Editio). New Jersey: Prentice-Hall.

Hebb, D. (1949). *The Organization of Behavior*. (W. Press, Ed.). New York.

Hodneland, E. (2003). *Segmentation of Digital Images*. University of Bergensis.

Kale, A. J. (2015). A Road Sign Detection and the Recognition for Driver Assistance Systems, (Icesa), 69–74.

Kobayashi, M., Baba, M., Ohtani, K., & Li, L. (2015). A Method for Traffic Sign Detection and Recognition Based on Genetic Algorithm. *2015 IEEE/SICE International Symposium on System Integration (SII)*, 455–460. https://doi.org/10.1109/SII.2015.7405022

Lim, J. S. (1990). *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice Hall.

Mehrotra, K., Mohan, C., & Ranka, S. (2001). *Elements of Artificial Neural networks*.

Orr, G., Schraudolph, N., & Cummins, F. (1999). No Title. Retrieved February 28, 2017, from https://www.willamette.edu/~gorr/classes/cs449/momrate.html

Roberts. (2015). Neural Networks. Retrieved February 25, 2017, from http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history2

Santra, A. K. (2013). *Denoising Images Under Multiplicative Noise*. National Institute of Technology, India.

Seiffert, U. (2002). Artificial Neural Networks on Massively Parallel Computer Hardware. In *European Symposium on Artificial Neural Networks Bruges* (pp. 319–330). Belgium.

Sekeroglu, B., & Khashman, A. (2004). *Intelligent Banknote Identification System*. Near East University.

Wu, Y., Liu, Y., Li, J., Liu, H., & Hu, X. (2016). Traffic Sign Detection based on Convolutional Neural Networks. *Proc. IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 201–206. https://doi.org/10.1109/IJCNN.2013.6706811

Wu, Y., & Zusheng, C. (2016). A Detection Method of Road Traffic Sign Based on Inverse Perspective Transform. In *ICOACS2016* (pp. 293–296).

Yang, Y., Luo, H., Xu, H., & FuchaoWu. (2016). Towards Real-Time Traffic Sign Detection and Classification. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *17*(7), 2022–2031.

Zhe, X., Ren, J., & Bao, C. (2016). A Traffic signs' detection method of contour approximation based on concave removal. *Proceedings of the 28th Chinese Control and Decision Conference, CCDC 2016*, *2*, 5199–5204. https://doi.org/10.1109/CCDC.2016.7531927

Zurada, J. (1992). *Introduction to Artificial Neural Systems*. (P. W. Company, Ed.). Minnesota.

APPENDICES

APPENDIX 1

PROGRAM LISTING

Main Function

```
clear
 clc;
sign=13;
photos=16;
train_pic=8;
path=cd;
addpath(path);

matrix =im_read_seg('Segmented Signs',sign,photos,40,'nshow',0.5);
%[matrix segmented]=im_read('signs',sign,photos,40,'nshow',0.5);
T=[];
for i=1:16
    T=[T eye(sign)];
end

 neural=newffunction(matrix,T,[50
50],{'tansig','tansig','tansig','tansig'},'traingdx',100,25000,100e-
6,0.01,0.09,0.5,0.5,0 );
 %neural=newffunction(matrix,T,[350 300
25],{'tansig','tansig','tansig','tansig'},'traingdx',100,25000,15e-
6,0.8,0.2,1,0.4,0.2);

 [neural tr]=train(neural,matrix,T);
 train_input=matrix(:,tr.trainInd);
 T_train = T(:,tr.trainInd);
 test_input = matrix(:,[tr.testInd tr.valInd]);
 T_test = T(:,[tr.testInd tr.valInd]);
 exam1=sim(neural,train_input);
 exam2=sim(neural,test_input);
% clear('DDF','IPF','OPF','PF','T','i','photos','sign','train_pic');
 XX1=(exam1>0.2);
 XX2=(exam2>0.2);
 EX1=abs(exam1-T_train);
 EX2=abs(exam2-T_test);
 o=0;
 tr_thr = 0.4;
 te_thr = 0.6;
 trRecInd=[];
 testRecInd=[];
 for i=1:length(EX1)
     if(all(EX1(:,i)<tr_thr))
         o=o+1;
         trRecInd(o)=i;
     end
 end
 aa1 = o ;
 o=100*o/size(train_input,2);
 ratio = sprintf('%0.4g',o);
 pr = strcat(['Training rate is ',ratio,' %']);
 disp(pr);
 disp(' ')

  o=0;
 for i=1:length(EX2)
    if(all(EX2(:,i)<te_thr))
         o=o+1 ;
         testRecInd(o)=i ;
    end
 end
 aa1 = aa1 + o;
 o=100*o/size(test_input,2);
 ratio = sprintf('%0.4g',o);
 pr = strcat(['Test rate is ',ratio,' %']);
 disp(pr);
 disp(' ');
 o = 100 * aa1 / 208;
 ratio = sprintf('%0.4g',o);
 pr = strcat(['Total rate is ',ratio,' %']);
 disp(pr);
 disp(' ');
```

Function `im_read_seg`

```
function matrix=im_read_seg(file,no_signs,no_photos,N,X,delay)
cd(file)
for i=1:no_signs
    cd (sprintf('%02s',int2str(i)));
     for j=1:no_photos
        a=imread([sprintf('%02s',int2str(j)),'.jpg']);

        if(j==1 && (strcmp(X,'show') || strcmp(X,'SHOW')))
            imshow(a);
            pause(delay);
        end
        b=imresize(a,[N N]);
        a=reshape(b,[N*N 1]);
        ve=double(a)./255;
        matrix(:,i + no_signs*(j-1))= ve ;
     end
    cd ..
end
cd ..
end
```

APPENDIX 2

OUTPUT OF ANN

Table 1: Training results of first signs sample (structure 1)

0,998	**0,004**	**0,000**	**0,002**	**0,002**	**0,000**	**0,004**	**0,002**	**0,001**	**0,985**	**0,001**	**0,000**	**0,005**
0,000	0,004	0,002	0,002	0,002	0,002	0,003	0,001	0,001	0,000	0,000	0,003	0,004
0,004	0,004	0,005	0,001	0,003	0,000	0,007	0,002	0,000	0,003	0,989	0,004	0,003
0,004	0,996	0,006	0,000	0,002	0,001	0,002	0,001	0,000	0,002	0,002	0,009	0,001
0,000	0,004	0,995	0,003	0,002	0,004	0,001	0,000	0,001	0,000	0,003	0,990	0,003
0,002	0,003	0,002	0,003	0,002	0,001	0,001	0,001	0,002	0,002	0,001	0,002	0,003
0,002	0,001	0,003	0,001	0,003	0,000	0,004	0,000	0,003	0,001	0,001	0,003	0,996
0,002	0,001	0,004	0,997	0,000	0,000	0,003	0,002	0,002	0,004	0,002	0,006	0,001
0,002	0,002	0,001	0,001	0,995	0,002	0,005	0,002	0,002	0,003	0,004	0,001	0,003
0,001	0,003	0,005	0,001	0,002	0,996	0,004	0,001	0,001	0,001	0,002	0,005	0,000
0,002	0,004	0,002	0,001	0,003	0,004	0,993	0,002	0,002	0,005	0,008	0,001	0,003
0,003	0,003	0,000	0,003	0,005	0,001	0,002	0,999	0,001	0,005	0,001	0,000	0,001
0,003	0,002	0,002	0,003	0,002	0,003	0,002	0,001	0,998	0,007	0,000	0,003	0,002

Table 2: Test results of first signs sample (structure 1)

0,000	**0,001**	**0,006**	**0,006**	**0,013**	**0,981**	**0,004**	**0,004**	**0,997**	**0,003**	**0,007**	**0,002**	**0,001**
0,996	0,000	0,003	0,001	0,000	0,000	0,004	0,003	0,000	0,001	0,002	0,003	0,003
0,005	0,992	0,004	0,007	0,000	0,000	0,004	0,008	0,003	0,002	0,001	0,003	0,002
0,005	0,002	0,000	0,952	0,000	0,003	0,996	0,002	0,004	0,690	0,000	0,003	0,000
0,002	0,003	0,003	0,023	0,000	0,002	0,004	0,001	0,000	0,050	0,001	0,036	0,004
0,001	0,001	0,994	0,003	0,004	0,001	0,003	0,001	0,002	0,002	0,002	0,001	0,931
0,005	0,001	0,005	0,001	0,000	0,013	0,001	0,005	0,002	0,001	0,003	0,003	0,002
0,002	0,002	0,006	0,001	0,003	0,010	0,001	0,002	0,002	0,001	0,973	0,000	0,002
0,001	0,005	0,002	0,002	0,002	0,006	0,002	0,004	0,002	0,001	0,001	0,952	0,002
0,003	0,002	0,004	0,001	0,002	0,001	0,003	0,005	0,001	0,001	0,000	0,010	0,003
0,003	0,006	0,005	0,002	0,003	0,001	0,004	0,992	0,002	0,000	0,000	0,000	0,007
0,002	0,001	0,003	0,002	0,001	0,009	0,003	0,002	0,003	0,002	0,004	0,001	0,002
0,004	0,000	0,004	0,002	0,481	0,007	0,002	0,002	0,003	0,003	0,003	0,009	0,001

Table 3: Output of 10 arbitrary fed test images

0,005	**0,003**	**0,011**	**0,002**	**0,007**	**0,003**	**0,003**	**0,018**	**0,005**	**0,009**
0,992	0,011	0,002	0,004	0,006	0,012	0,004	0,014	0,989	0,006
0,008	0,004	0,016	0,008	0,975	0,004	0,000	0,001	0,008	0,967
0,004	0,011	0,005	0,006	0,008	0,010	0,002	0,019	0,004	0,006
0,005	0,011	0,007	0,002	0,006	0,011	0,001	0,004	0,006	0,005
0,006	0,000	0,004	0,005	0,004	0,000	0,005	0,002	0,006	0,003
0,007	0,989	0,005	0,008	0,007	0,990	0,004	0,008	0,009	0,006

0,010	0,005	0,004	0,008	0,001	0,004	0,001	0,004	0,010	0,001
0,001	0,004	0,005	0,991	0,012	0,004	0,007	0,001	0,001	0,012
0,006	0,002	0,009	0,004	0,004	0,002	0,967	0,006	0,007	0,003
0,002	0,010	0,969	0,008	0,032	0,010	0,003	0,000	0,003	0,031
0,011	0,006	0,001	0,004	0,002	0,006	0,006	0,819	0,012	0,002
0,001	0,008	0,019	0,004	0,004	0,007	0,016	0,011	0,002	0,004

Table 4: Output of 10 arbitrary fed test images

0,005	**0,003**	**0,011**	**0,002**	**0,007**	**0,003**	**0,003**	**0,018**	**0,005**	**0,009**
0,992	0,011	0,002	0,004	0,006	0,012	0,004	0,014	0,989	0,006
0,008	0,004	0,016	0,008	0,975	0,004	0,000	0,001	0,008	0,967
0,004	0,011	0,005	0,006	0,008	0,010	0,002	0,019	0,004	0,006
0,005	0,011	0,007	0,002	0,006	0,011	0,001	0,004	0,006	0,005
0,006	0,000	0,004	0,005	0,004	0,000	0,005	0,002	0,006	0,003
0,007	0,989	0,005	0,008	0,007	0,990	0,004	0,008	0,009	0,006
0,010	0,005	0,004	0,008	0,001	0,004	0,001	0,004	0,010	0,001
0,001	0,004	0,005	0,991	0,012	0,004	0,007	0,001	0,001	0,012
0,006	0,002	0,009	0,004	0,004	0,002	0,967	0,006	0,007	0,003
0,002	0,010	0,969	0,008	0,032	0,010	0,003	0,000	0,003	0,031
0,011	0,006	0,001	0,004	0,002	0,006	0,006	0,819	0,012	0,002
0,001	0,008	0,019	0,004	0,004	0,007	0,016	0,011	0,002	0,004

Table 5: Training results of the first configuration C. 1

S3	**S5**	**S10**	**S11**	**S12**	**S2**	**S3**	**S5**	**S9**	**S11**	**S12**
0,006	0,010	0,012	0,010	0,004	0,004	0,006	0,008	0,000	0,010	0,004
0,010	0,006	0,004	0,005	0,007	0,981	0,010	0,010	0,006	0,004	0,007
0,984	0,001	0,014	0,011	0,003	0,011	0,984	0,000	0,004	0,011	0,003
0,010	0,022	0,011	0,006	0,006	0,003	0,010	0,008	0,006	0,013	0,006
0,001	0,990	0,008	0,009	0,002	0,002	0,001	0,990	0,005	0,019	0,002
0,012	0,010	0,003	0,000	0,008	0,002	0,012	0,009	0,001	0,001	0,007
0,003	0,008	0,001	0,002	0,007	0,018	0,003	0,018	0,010	0,002	0,006
0,007	0,002	0,007	0,007	0,004	0,016	0,007	0,005	0,009	0,012	0,004
0,006	0,010	0,011	0,004	0,004	0,002	0,006	0,006	0,986	0,006	0,004
0,008	0,014	0,984	0,007	0,005	0,003	0,008	0,019	0,010	0,006	0,005
0,007	0,011	0,015	0,979	0,007	0,002	0,007	0,015	0,008	0,977	0,007
0,009	0,007	0,004	0,012	0,989	0,001	0,009	0,006	0,002	0,011	0,988
0,008	0,008	0,006	0,000	0,007	0,003	0,007	0,006	0,014	0,000	0,006

Table 6: Output results of the test of first configuration S.1

S1	S2	S4	S6	S7	S8	S9	S13	S1	S4
0,986	0,002	0,009	0,005	0,010	0,013	0,000	0,007	0,723	0,005
0,001	0,985	0,005	0,002	0,013	0,014	0,005	0,002	0,029	0,001
0,007	0,008	0,005	0,006	0,000	0,004	0,004	0,003	0,128	0,003
0,010	0,003	0,986	0,003	0,006	0,002	0,008	0,008	0,005	0,958
0,001	0,002	0,004	0,010	0,012	0,005	0,005	0,010	0,000	0,095
0,004	0,001	0,013	0,986	0,009	0,006	0,001	0,006	0,002	0,011
0,005	0,012	0,011	0,002	0,984	0,005	0,011	0,005	0,007	0,018
0,003	0,011	0,001	0,006	0,003	0,984	0,007	0,009	0,009	0,001
0,001	0,004	0,004	0,001	0,010	0,010	0,987	0,017	0,018	0,010
0,011	0,006	0,001	0,003	0,002	0,002	0,009	0,007	0,027	0,003
0,002	0,002	0,006	0,010	0,002	0,012	0,007	0,002	0,004	0,008
0,008	0,003	0,004	0,008	0,005	0,005	0,002	0,003	0,001	0,002
0,007	0,001	0,002	0,007	0,013	0,010	0,016	0,992	0,060	0,010