ALI IDRESS RAHEL BUGHNEDA

ANN FOR LEAF SPECIMEN CLASSIFICATION

NEU 2018

ANN FOR LEAF SPECIMEN CLASSIFICATION

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

By
ALI IDRESS RAHEL BUGHNEDA

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

NICOSIA, 2018

ANN FOR LEAF SPECIMEN CLASSIFICATION

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

By
ALI IDRESS RAHEL BUGHNEDA

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

NICOSIA, 2018

Ali Idress Rahel BUGHNEDA : ANN FOR LEAF SPECIMEN CLASSIFICATION

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 Master of Science in Electrical & Electronic Engineering

Examining Committee in Charge:

Assist.Prof. Dr. Eser Gemikonekli	Committee Chairman, Department of Computer Engineering, UOK
Assist.Prof.Dr. Yoney Kirsal Ever	Department of Software Engineering, NEU
Assoc.Prof.Dr. Kamil Dimililer	Supervisor, Department of Electrical and Electronic 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: Ali Bughneda

Signature:

Date:

ACKNOWLEDGEMENTS

I am happy to be in the end point of this thesis work and to be able to write these words. First of all, I would like to thank my great supervisor; Assoc. Prof. Dr. Kamil Dimililer for his patience and support. I would also like to thank all the members of my family.

Last but not least, I would like to express all my sincere feelings to my friends and university colleagues for the times we have spent together. I also express my deep feelings to the Professors, doctors and staff members at Near East University for their continuous support.

To my parents and family….

ABSTRACT

Leaf is a part of the vascular plant that represents principal lateral adjunct of the plant trunk. Plant leaf contains multiple features that create the difference between plant and another plant. Main features of the leaf are the margins, shape, and structure of the leaf. Each plant leaf has its different features that are extinctive from other plants' leaves. The classification of plants based on their leaves is used since long time by humans. This work focuses on the classification of plants leaves using computerized approaches and neural networks. The work will consider the segmentation of the plant leaves prior to the application of neural networks classifiers to classify leaves into sub-categories. Beginning with the final representation of leaves, the work will apply the neural network algorithms to classify different plant leaves. The algorithm will initially train the neural network using large number of available images. Upon the end of the training process of the proposed system, a test process will start using other images to clarify whether the system is learned or not. Different comparisons will be applied during the work to verify the effect of different parameters and processes on the classification quality.

Keywords: Artificial neural networks; back propagation; Leaf recognition; plant classification; segmentation

ÖZET

Yaprak, bitki gövdesinin temel lateral tamamlayıcısını temsil eden damarlı bitkinin bir parçasıdır. Bitki yaprağı, bir bitkiyle diğer bitki arasındaki farkı oluşturan çeşitli özelliklere sahiptir. Yaprağın temel özellikleri kenarlar, şekil ve yaprak yapısıdır. Her bitki yaprağı diğer bitki yapraklarından farklı, kendine ait özellikler taşımaktadır. İnsanlar, uzun bir süredir, bitkileri yapraklarına göre sınıflandırmaktadır. Bu çalışma, bitkilerin bilgisayar donanımlı yaklaşımlar ve nöral ağlar kullanılarak yapraklarına göre sınıflandırılması üzerine yoğunlaşmaktadır. Çalışmada, yaprakları alt kategorilere göre sınıflandırmak için nöral ağ sınıflandırılması uygulanmadan önce bitki yapraklarının segmentasyonu değerlendirilecektir. Çalışma yaprakların son hallerinin gösterilmesiyle başlayarak, farklı yaprak çeşitlerinin sınıflandırılması için yapay nöral geri yayılma uygulanacaktır. Algoritma, ilk olarak, mevcut olan resimleri kullanarak nöral ağ yapısını öğretecektir. Sunulan sisteme ait eğitim sürecinin sonuna doğru, sistemin öğrenilip öğrenilmediğini belirlemek için farklı resimler kullanılarak bir test süreci başlatılacaktır. Sınıflandırma kalitesine ilişkin olarak farklı parametre ve süreçlerin etkisini doğrulamak için çalışma boyunca farklı karşılaştırmalar yapılacaktır.

Anahtar kelimeler: Bitki sınıflandırması, geri yayılma, yapay nöral ağlar, yaprak tanıma

TABLE OF CONTENTS

CHAPTER 4: RESULTS AND DISCUSSIONS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN: Artificial neural networks

BP: Back propagation

LR: Learning rate

MF: Momentum factor

CHAPTER 1

INTRODUCTION

1.1 Introduction

Identification of plant species using their leaves and stems is a task that needs special experience. Manual identification of plant species is a very difficult task that requires a lot of experience in addition to huge books that shows the different types of plants and their leaves. However, the increasing capabilities of the new cameras and mobile phones gives opportunity for picturing plant leaves and using automatic ways to identify them. The aim is give the opportunity for people to be able to identify all types of plants in real life using the power of their mobiles' cameras. A small application that is equipped with a well trained algorithm can be able to offer such a great job and to make people avoid many losses due to bad identification of plants, especially, the poisonous plants.

Plant identification is very important also to help in the development of automated farms where machines can do the whole job (Kadir, Nugroho, Susanto, & Santosa, 2013). Planting, cleaning, and harvesting the crops using automated machines is no longer science fiction. Many developed countries use machines nowadays in harvesting the agriculture products. These automated machines need to identify the crop and separate the wrong parts or grasses while harvesting. This classification needs to be improved to give the best results without any human intervention.

Different approaches use segmentation methods in the identification of plants leaves. One of the main disadvantages of these approaches is that any error in the segmentation will lead to a wrong identification of the subject leaf (Amin & Khan, 2013). The wrong segmentation can be the reason of overlapping leaves, sick plants, or broken parts of the plant leaves.

In this work, an approach is proposed for the identification of plant leaves based on image processing techniques and artificial neural networks. Image processing techniques help improving the quality of images to separate any noise or disturbances. The use of image

processing can increase the performance of the identification process and reduce the rate of error. After the application of image processing techniques, an artificial neural network will be applied on the images to train it for the different plant types. By the end of training process, the network will be ready to classify and identify the different types of leaves even if there were not presented before.

Artificial neural networks are human brain like networks. They have been implemented and used to imitate the shape and function of brain structure. Neural networks are very flexible and can change their structure to simulate the way of real brain. Many fields of science nowadays use the artificial intelligence of neural networks in their development.

1.2 Methodology of the Work

The proposed work aims to investigate the use of artificial neural network of many hidden layer in the classification of leaves specimen (Mallah, Cope, & Orwell, 2013). The work will start initially by collecting database of leaves images from different sources in RGB image form. The collected images will be then converted to gray images to simplify the computer processing tasks. The converted images will then be filtered and resized to be suitable for neural network applications. By the end of image processing phase, the neural network will be built and trained using a large amount of database images. Then the network will be tested and generalized.

1.3 Dataset Description

The dataset consists approximately of 761 images of leaf specimens for 19 different samples of plants. These images were all taken as JPG format. Figure 1.1 below shows an example of the captured images for one of the plants. This image will be treated using image processing methods and converted to gray scale. The next step will be to scale down this image so that it will be less size to be fed to neural networks.

Figure 1.1: Plant leaf image RGB

The flowchart below represents the steps proposed in this work to acheive the required goal of the work.

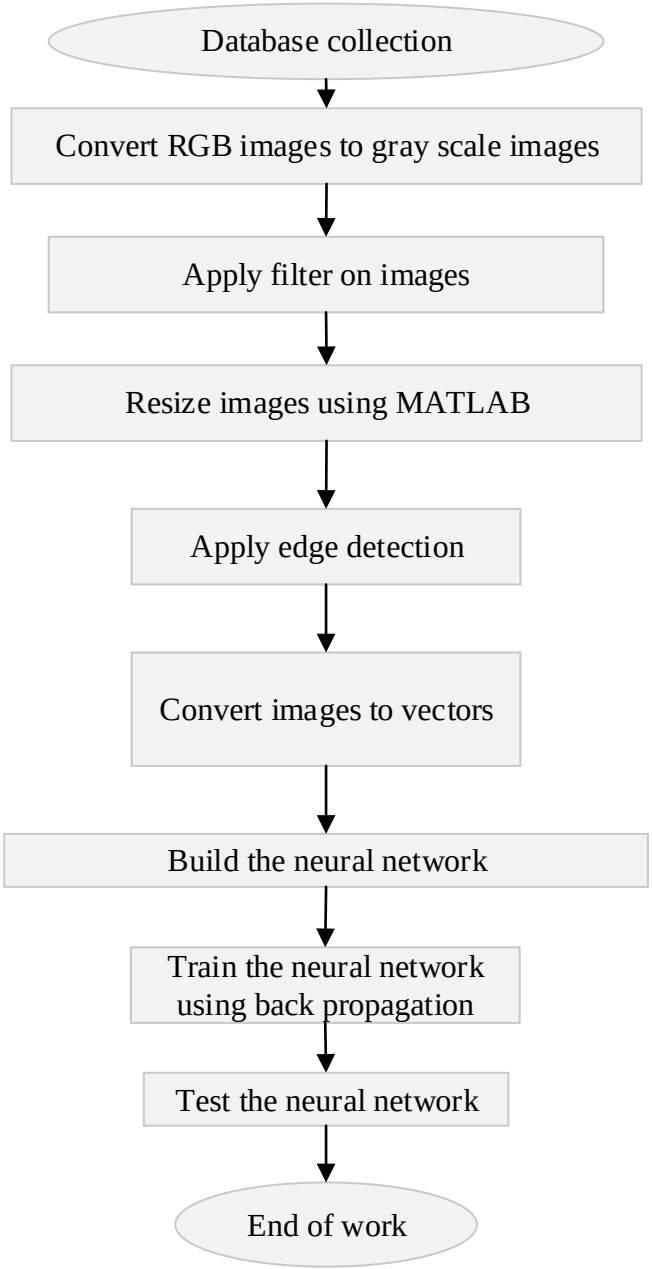


Figure 1.2: flowchart of the proposed work

1.4 Contribution of the Thesis

The work done in this thesis can be considered as a positive effort added to the previous works in the environmental studies. The proposed methods and application for ANN in the plant classification and recognition are promising for the automated recognition of different plant types including the rare and strange families of plants.

CHAPTER 2

IMAGE PROCESSING

2.1 Introduction

Image processing has been one of the important fields of science in the twentieth and 21st century. The use of image processing techniques and digital images has given a great revolution for the fields of communication and telecommunication. As a result of the revolution in the science of image processing, we can watch our TV's with high density videos and clear images. Internet images are also presented at very high resolutions with help of all great image processing techniques. In fact, all our life is surrounded by images.

Vision in human eye is nothing but a biological representation of an analogue image processing with all its detailed processes. The human eye is a great biological camera that has the ability to continuously capture infinite number of images, transfers them to the brain which can analysis and process these images. The brain then translates these images into meaningful things for us. The digital image processing is the digital counterpart of the analogue image processing in the human (Gonzalez & Woods, 2001). Digital image processing; abbreviated mostly DIP, is the mathematical processing of images that are represented digitally using computers. The fields of DIP applications are very vast and can't be easily covered in one topic. In this work, some of the important DIP processes will be presented and discussed briefly preparing to be introduced in the practical application for leaf recognition. The discussion will cover mainly the image representation, image filtering, image segmentation, and image conversion processes. All these topics will be covered in this chapter in order to give an idea about the proposed work.

2.2 How Images are Represented

When you capture an image using your camera, mobile or computer; the machine captures the analogue information of the image and translates it to tinny blocs of digital values. These tinny blocs that contain the digital values are called pixels. The collection of pixels

constructs all types of the digital images. Digital images are represented using different methods and can be classified as follow:

1- 1 bit images that can contain binary values like 0 and 1 in each pixel. These images contain tow colours and can give fewer details. Such images are famous in some arts where details need to be defined sharply or some medical images where sharp details are needed.
2- 8 bit images where each pixel value is defined using an 8 bit number. This type is known also as indexed colour (Sekeroglu & Khashman, 2004). Gray scale images and most of internet images are main examples of indexed colour images. Each pixel in an 8 bit images can contain 256 different colour levels.
3- 16 bit images where each pixel is represented using word space in the computer. 2 bytes of data are reserved for each pixel in this type of images. This type of images is normally known by high colour images (Sekeroglu & Khashman, 2004). This is due to the fact that it contains millions of colour grades.

The digital image is then represented in a structure of blocs of data arranged such that they show colour grades in the frame. Images can be stored in different formats as it is well known. RAW images, JPEG, PNG, JIF, and BMP images are all types of images represented using different compression methods (Boran Sekeroglu, 2004).

Figure 2.1: JPEG image of leaf

Figure 2.1 above shows the JPEG image version of a plant leaf. Jpg image is a compressed version of raw image. Figure 2.2 presents the indexed version of the JPG image shown above. The binary image of the leaf is generated and shown in Figure 2.3 below. Finaly, the gray scale version of the image is represented in Figure 2.4. it can be considered as the best choice for ANN application.

Figure 2.2: Indexed version of the leaf image

Figure 2.3: Binary version of leaf image

Figure 2.5 shows the difference between pixels in the binary image and the gray scale image. Gray scale image can contain more details about each pixel of the image and offer a description of the grades of light in it.

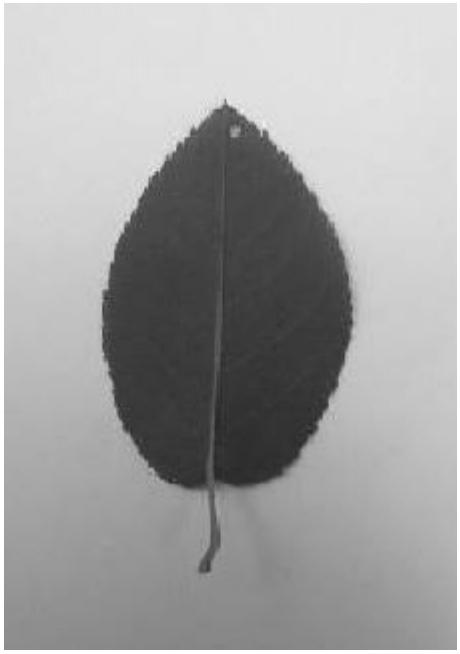


Figure 2.4: Gray scale version of the leaf image

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

191	191	188	187	85	74	68	65	65	66	65	65	64	63	62
192	193	189	184	97	71	67	63	63	64	65	65	63	62	61
193	197	185	187	81	77	70	75	70	65	63	63	64	64	63
191	193	186	186	122	80	66	70	68	65	64	65	66	64	63
193	190	185	173	141	78	68	65	65	65	66	67	66	64	62
198	188	181	159	112	75	73	65	65	65	65	65	65	63	62
199	184	182	172	78	80	72	67	66	65	64	62	62	62	63
195	181	187	196	69	86	66	66	66	65	63	61	60	61	63
190	188	186	196	71	80	67	62	64	65	65	62	61	61	62
188	199	181	176	67	67	73	57	61	65	66	65	62	62	62
191	190	179	186	82	76	56	71	65	62	64	64	59	56	57
187	190	182	182	92	71	67	68	63	61	64	64	61	59	60

Figure 2.5: (a) Binary image pixels vs. (b) gray scale image pixels

2.3 Conversion from Coloured Image to Gray Scale Image

The conversion from RGB image to gray scale image is the process of representing the coloured image in form of the scale of white and black image. The gray scale image

contains all information of the image except from the colours which are less important in computer applications. Gray scale images remind us with the old white and black TVs in the beginning of the 90s of the last century. Although the formula (1.1) looks logical for the conversion from RGB to gray scale, however, the researches show that the human eye has different sensitivity levels for each one of the main colours. The formula (1.2) was found to be more realistic in the conversion between the RGB and the gray images (Zollitch, 2016).

$$G_s = \frac{R}{3} + \frac{G}{3} + \frac{B}{3} \quad (2.1)$$

$$G_s = \frac{R}{3.34} + \frac{G}{1.7} + \frac{B}{8.77} \quad (2.2)$$

Where, G_s refers to the gray scale image, R refers to the red colour, G refers to the green colour, and B refers to the blue colour (Santra, 2013). The next figure shows the RGB image along with the gray scale images created using the two previous formulas.

(a) RGB image

(b) Gray scale image 1

(c) Gray scale image 2

Figure 2.6: RGB image vs. gray scale images using the two previous formulas

2.4 Edge Detection in Digital Images

Edge detection is the process in which significant object properties in digital images are being detected and captured. These properties include discontinue lines in the characteristics of the object. The main aim of edge detection is to identify and separate these variations in the property of objects within the image. Efficient edge detection is able to focus on the important objects or regions of interest in the images. it is very useful to detect and identify the ROI's in an image. To perform better edge detection the derivative of the image pixel is calculated. However, derivatives are subject to be affected by different types of noise of different sources, hence; smoothening the image prior to the application of derivative is a must to improve the quality of image edge detection.

Edge detection is a very difficult and sensitive process in image processing. Each different application requires different edge detection method that is designed to accomplish its task. in this work, different general famous edge detection methods will be discussed briefly and explored. Famous edge detection methods like Sobel, Canny, Prewitt and other edge detection methods will be presented. Results of all discussed methods will be also presented and compared.

2.4.1 Sobel edge detection

Sobel operator applies a two dimensional space gradient calculation to highlight the high frequency regions in the special domain. It finds the gradient of each pixel in gray scale images. Sobel edge detection method employs two kernels of 3*3 matrixes. The two matrixes are shown as follow (Robert, Simon, Ashley Walker;, & Wolfart, 2003):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$G_y = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.3}$$

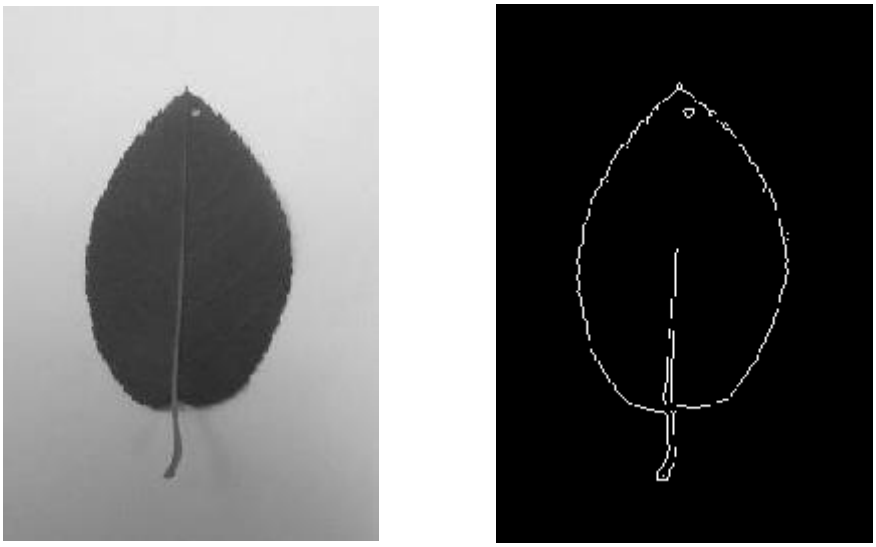
These two matrixes are constructed specifically to respond the best for image variations in the horizontal and vertical directions. The two kernels are applied separately to ensure the calculation of the horizontal and vertical gradients. The two gradients are then used to find the absolute gradient value of the image. The magnitude of the gradient of the image can be found by (Robert et al., 2003):

$$/G/ = \sqrt{G_x^2 + G_y^2} \tag{2.5}$$

Where; the calculation of the gradient magnitude can be costly and slow, therefore, an approximation of the gradient can be accepted in a faster manner without affecting the final result:

$$/G/ = /G_x/ + /G_y/ \tag{2.6}$$

Sobel edge detection is less affected by the noises that appear in the image.



(a) Gray scale image (b) Sobel edge image

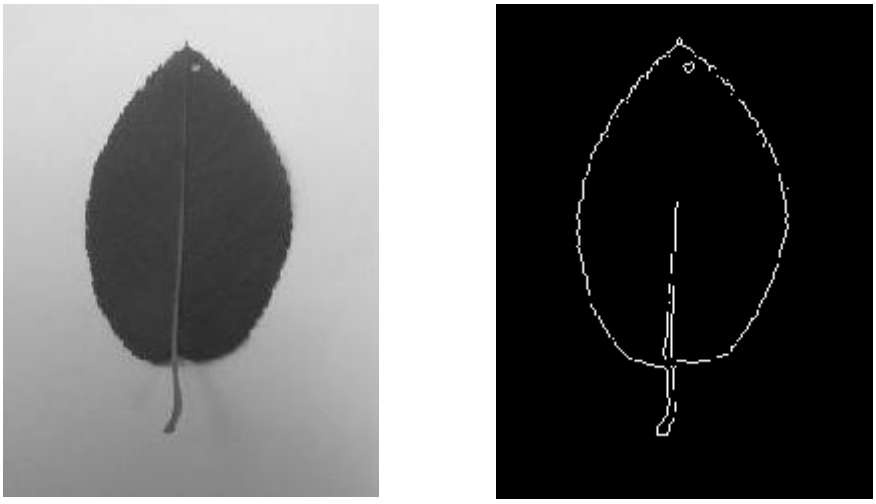
Figure 2.7: Application of Sobel operator for leaf image segmentation

2.4.2 Prewitt edge detection

Prewitt edge detection is similar to the Sobel method as it also uses similar kernel to find the gradient in an image. The kernel in Prewitt method is isotropic as shown in the next equations defining the horizontal and vertical kernels (Chaple, Daruwala, & Gofane, 2015).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$G_y = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.3}$$



(a) Gray scale image (b) Prewitt edge image

Figure 2.8: Prewitt edge detection algorithm

2.4.3 Canny edge detection

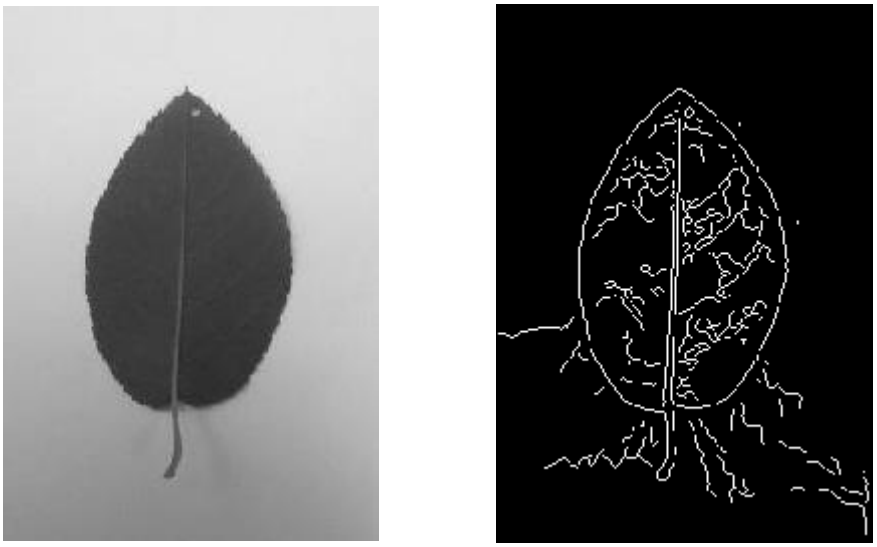
Canny edge detection is one of the famous edge detection and segmentation methods. Canny segmentation method is the distinction of parts of image into areas. It highlights the important areas of the image to identify these areas. The feature areas are very important for processing targets. The edges are defined in most of cases in term of zones where the pixels change sharply. Canny edge detector is a common edge detection algorithm. This

algorithm is employed to increase the signal to noise ratio and identify the different features of the image. Generally, Canny edge detection algorithm is composed of different steps (Shen & Tang, n.d.).

At the beginning, the image is smoothed to remove any kind of noises or blur from it. This is then an application of filter to the image. Filtering process is carried out by convolving a Gaussian filter with the image. After filtering the image, the gradient of each column and line of the image is found. The more the change in the image pixels the higher the gradient is. This fact is used to identify the sharp changes in the mage pixels. The gradient of a pixel in the image can be given using the next formula:

$$G_{xy} = \sqrt{G_x^2 + G_y^2} \qquad (2.9)$$

Where; Gx and Gy are the horizontal and vertical gradients of the image. Canny edge detection uses threshold to eliminate the unwanted regions of the image and highlight the regions of interest.



(a) Gray scale image (b) Canny edge image

Figure 2.9: Application of Canny edge detection

2.4.4 Roberts edge detection

The Robert operator is used to approximate the gradient of image using discrete differentiation. Robert mask is a 2*2 matrix that convolve with the whole image using its horizontal and vertical versions (Chaple et al., 2015). These masks give the horizontal and vertical derivative approximation of an image. Suppose the image matrix shown in next equation, and the two masks of Roberts.

$$I = \begin{bmatrix} Z_1 & Z_2 \\ Z_3 & Z_4 \end{bmatrix} \tag{2.10}$$

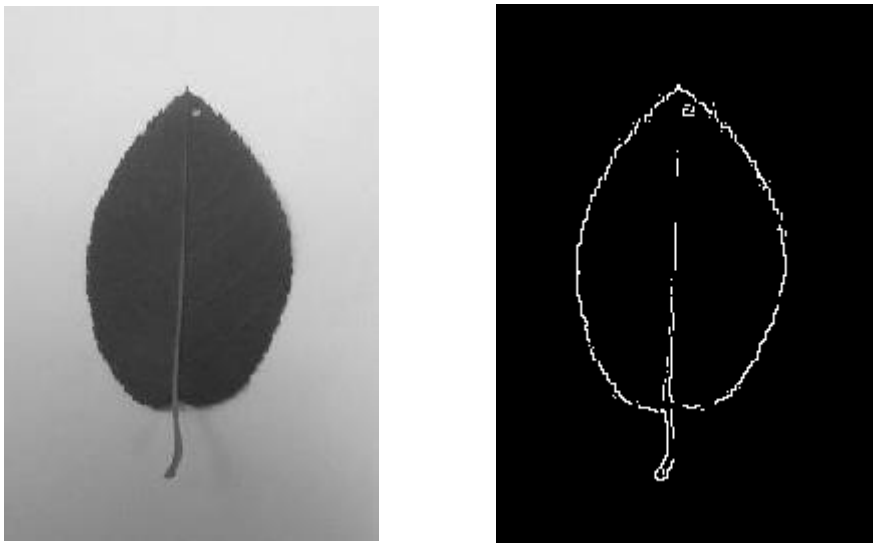
$$H.mask = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.11}$$

$$V.mask = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{2.12}$$

From the image matrix and the Roberts masks, the horizontal and vertical derivative approximations can be given by (Chaple et al., 2015):

$$\begin{aligned} G_x &= Z_4 - Z_1 \\ G_y &= Z_3 - Z_2 \end{aligned} \tag{2.13}$$

At this stage, a threshold is applied to the image to highlight the high gradient values and eliminate the unwanted image regions. Figure 9 below shows the results of applying Roberts edge detection algorithm on the leaf images.

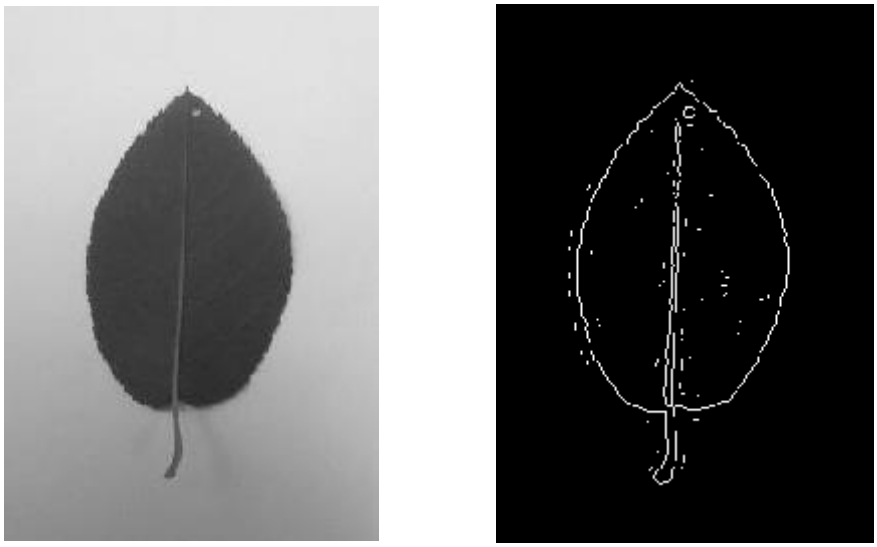


(a) Gray scale image (b) Roberts edge image

Figure 2.10: Roberts edge detection

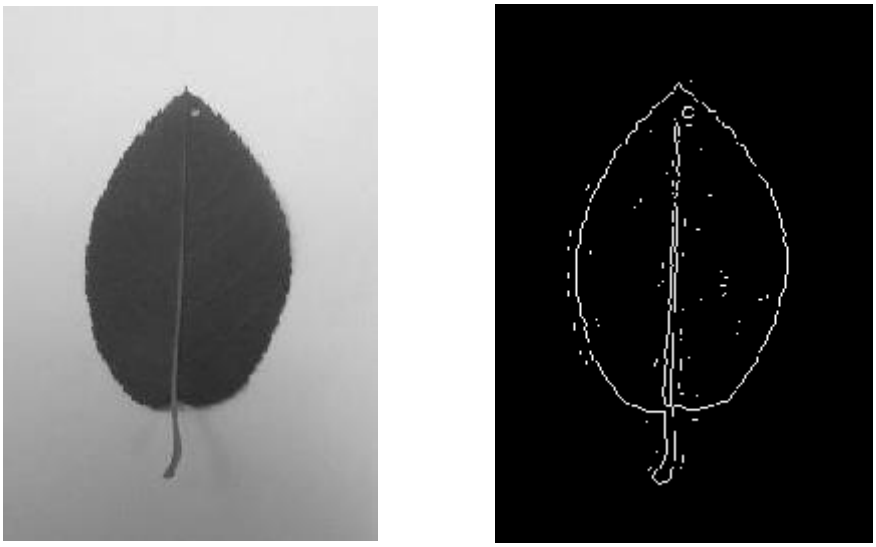
2.4.5 Logarithmic and zero cross edge detection method

These are the last two algorithms applied in our work for image segmentation. Application of these two methods has given the images shown in the next two figures.



(a) Gray scale image (b) Logarithmic edge image

Figure 2.11: Logarithmic segmentation results



(a) Gray scale image (b) Zerocross edge image

Figure 2.12: Zero cross segmentation process

2.5 Wiener Filter

Wiener filter is an adaptive filter type which uses the error minimization formulas to reject the noise from images. It is based on the mean squared error approximations. It is considered as a frequencies filter that can be used with discrete Fourier transform of the image. The convolution of the Wiener filter with the image can generate the spectrum of the image. The reconstruction of the image is carried out by applying the inverse version of discrete wavelet transform. (Khajwaniya & Tiwari, 2015; Wang, Peng, Wang, & Peng, 2015) have proposed Wiener filter as a best fit for the image filtering and restoration in different applications (Mohan, Mridula, & Mohanan, 2016).

Wiener filter is a type of low pass filters that can work with gray scale images. It shows high efficiency in the removal of additive noise types from these images. Wiener filter finds some statistical information around each pixel in the image and generate an estimation of the correct new value of the pixel based on these statistical results. The mean of the Wiener filter is given by:

$$m = \frac{1}{MN}\sum_{i,j}^{M,N} p(i,j) \qquad (2.14)$$

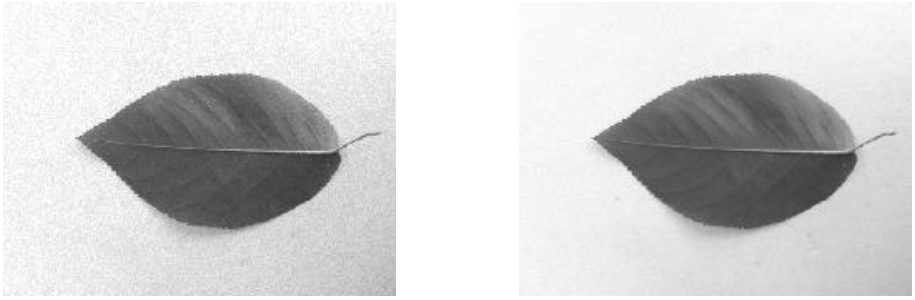
Where M and N are the dimensions of the Wiener window, “p” is the pixel value, “m” is the mean value of the window pixels. The standard deviation is also a well known term in statistics that is given by:

$$\sigma^2 = \frac{1}{MN}\sum_{i,j}^{M,N} p^2(i,j) - m \qquad (2.15)$$

The new pixel value is then generated with help of the variance of the noise that is either estimated or supposed. The new value is given by:

$$P_{new}(i,j) = m + \frac{\sigma^2 - V^2}{\sigma^2}\,(/p\ - m/) \qquad (2.16)$$

Where; the term “v” refers to the noise variance. The new pixel value is a filtered version of the original pixel value. Wiener filter has proven high ability to remove additive noise types and restore the original images. Figure 2.13 below presents the result of applying Wiener filter on the noisy image “a”. The figure shows that the filter has reduced the noise in the area of interest in the image (leaf).



(a) Gray noisy image (b) Wiener filtered image

Figure 2.13: Application of Wiener filter to noisy image

CHAPTER 3

ARTIFICIAL NEURAL NETWORKS

3.1 Overview of the Chapter

In this chapter, the concept of artificial neural networks will be discussed and the basic elements of the neural network structure will be discussed as well. Many artificial neural networks related topics will be highlighted in this chapter such as the similarity between human brain and artificial networks, learning in artificial neural networks, and different mathematical representation of transfer function.

At the end of this chapter, back propagation algorithm which is considered as one of the most effective learning algorithms will be presented and discussed.

3.2 Introduction

The human brain is one of the most important organs that is characterized by complexity. This organ performs many important and accurate tasks such as decision making, learning, logical thinking, and memorizing information (Mehrotra, Mohan, & Ranka, 2001).

Scientists studied the human brain accurately and tried to understand the principle of the work of nervous system, especially in the process of learning and decision-making. Historically it can be said that artificial neural networks started as a paper by McCulloch and Pitts in 1943, they provided a formal mathematical model describing the simplest structure of a human brain, in 1949 Hebb wrote a paper in which the Hebbian learning rule was proposed, this paper became one of the cornerstones for the development of training algorithms in the field of neural networks. The first perceptron model which is consider as the simple single layer networks was introduced by Rosenblatt in 1958 and based on this model, Werbos introduced his model as the three-layered perceptron network and wrote about back propagation algorithm in 1974. However in 1982 Hopfield published a series of papers on Hopfield networks, while Kohonen developed a totally unique kind of network model so-called self-organizing maps in the same year (Cios & Shields, 1997).

The research on artificial neural networks has remained active which led to the emergence of many developments in networks such as the sub-field of radial basis function networks in 1990s. However, in the current decade the power of neural networks is apparent, leading to many new network types, such as hybrid algorithms and hardware for neural information processing (G. Anderson & McNeill, 1992).

In the following sections the idea of neural networks will be adequately explained where the artificial neural networks will be defined and compared with the principle of the functioning of the nervous system in human. The most important components of the networks such as the concept of weights, layers and transfer function will be clarified.

3.3 Artificial Neural Networks Analogy

Artificial neural network is a computer technology that attempt to build models which are biologically inspired rather than an exact copy of how the brain is working.

In order to understand the idea of artificial neural networks, a general overview of the real human brain will be explained. Human brain is a huge network consisting of thousands of billions of neurons which are interconnected in a complex network in order to perform all complex tasks, Figure 2.1 demonstrates a biological neural network which consist of a simplified form of two neurons, each neuron consists of soma, dendrites, axon and synapse (D. Anderson & McNeill, 2010).

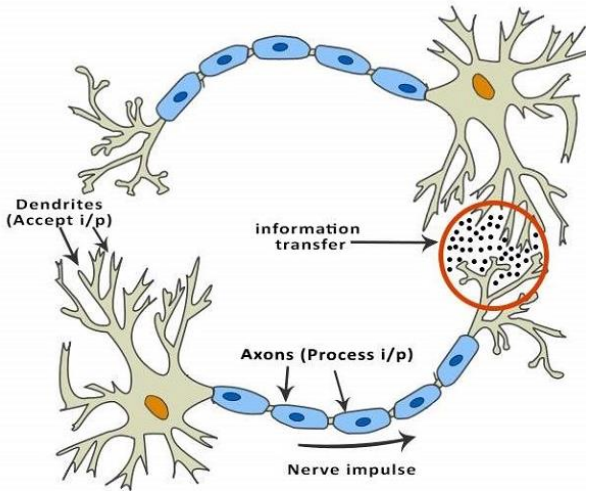


Figure 3.1: A schematic diagram of biological neurons

Dendrites receive signals from other neurons or the external environment, to the cell body; whereas, axons act as the output of neuron which take signals away from the cell body. However, neurons do not physically touch each other, the spaces between the sending neuron and the receiving neuron are called synapses , this synapses is permit a neuron to pass an electrical or chemical signal to another neuron or to the target. Moreover, the synaptic connections between cells are malleable and constantly changing.

The Figure 2.2 shows the similarities between the structure of the biological neuron and the artificial neuron, where it can be considered that the soma is similar to artificial neuron in the mechanism of action while dendrites act as inputs in artificial neuron. Moreover it can be considered that the function of the axon is similar to output function, whereas, the synapses in biological neurons simulate the weights in artificial neural network (G. Anderson & McNeill, 1992).

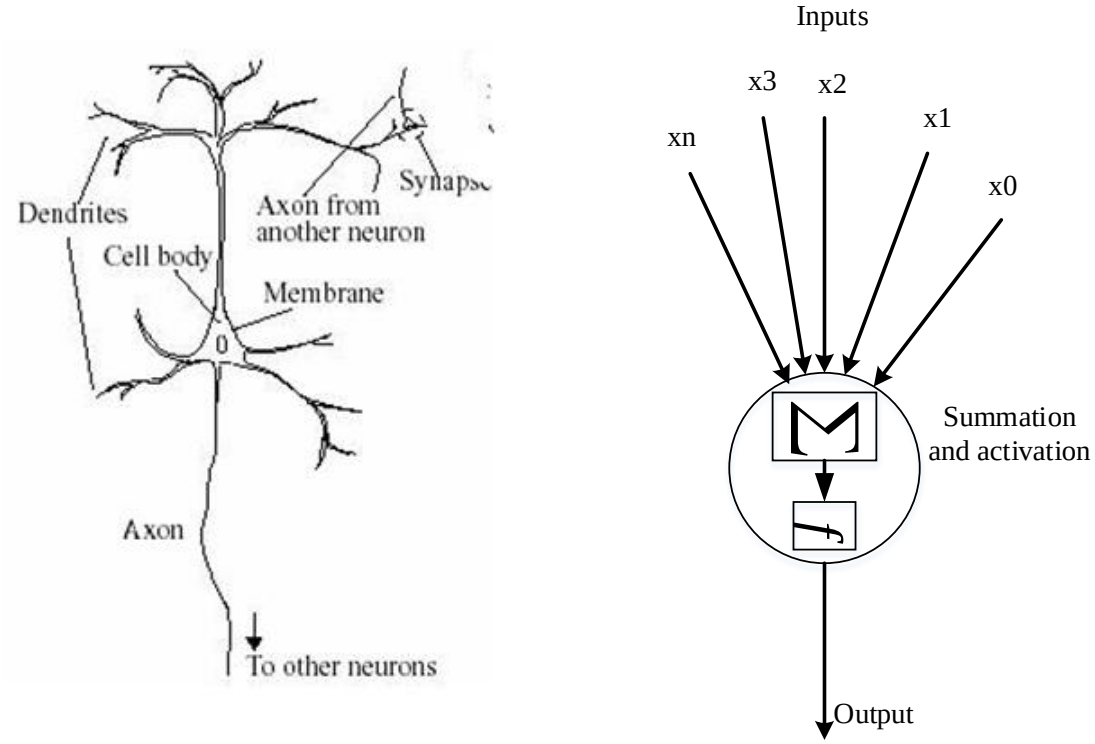


Figure 3.2: The similarity between biological neuron and artificial neuron

The biological neuron receive many signals from neighboring neurons through the dendrites, this signals reach the neuron's body only if it has enough energy to activate it. The same way in which artificial neurons work, the neuron receive the input signals and

each input signal is linked to its own weight that reflect the power of signal to activate the output of neuron; otherwise, the output from the neuron will not be generated and transmitted to the next neuron.

3.4 Artificial Neural Network Structure

Artificial neural networks (ANNs) which are inspired by biological neural networks are considered as learning models consisting of interconnected neurons, these connections between neurons have numeric weights and responsible for activating neuron output by multiplying its value with input signals (A.D.Dongare, R.R.Kharde, & D.Kachare, 2012).
In general, it can be said that ANNs consist of processing units called neurons, theses neurons are distributed within the system structure in the form of layers. The following sections will explain the most important components of the artificial neural network structure.

3.4.1 The layers

In artificial neural network, many different models and designs can be formed depending on the number of neurons and the number of layers in addition to the number of neurons within a single layer. Typically, the network consists of input layer, output layer and at least one hidden layer.

In the input layer the received data will be transferred to the first hidden layer, by using mathematical operation the data will be processed and transferred to the next hidden layer.

The simple model of the neural network consists of the input layer responsible for receiving the input signals; this layer passes the information coming from the surrounding environment to the next layers in the networks without performing any mathematical operations on that data (Seiffert, 2002).

The hidden layers are named because they do not have a direct contact with the environment surrounding the network, these layers perform computations and transfer information that received from the input layer and then passed to the next layer in the network. However, the number of hidden layers in the network varies depending on network's design, where in feed-forward network is possible to have one hidden layer at least or zero hidden layer.

The final layer which is responsible for delivering information to the outside world is called the output layer.

In feed-forward networks where data flows in one direction, from input to output layer without any loops or reverse flow, the networks are divided according to the number of layers into single layer perceptron and multi-layer perceptron.

3.4.2 Single layer perceptron

The single layer perceptron is the simplest model in ANNs, Figure 2.3 illustrate the structure of single layer perceptron which is consist of input layer connects directly to the output layer where there is no hidden layers in this type of feed-forward network.

In SLP, the neuron is connected typically to all input signals, however all neurons is allocated in parallel form in order to create single layer perceptron model (Colin, 1996).

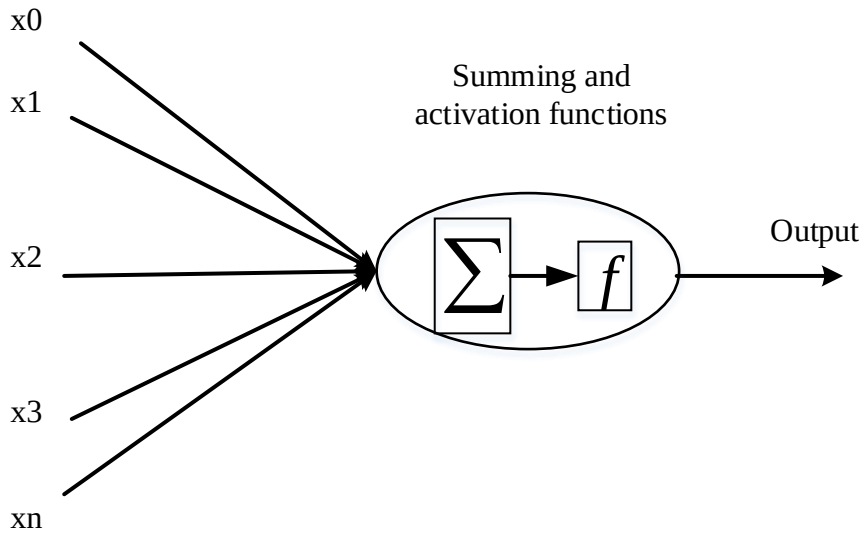


Figure 3.3: structure of single layer perceptron (SLP)

Mathematically, assuming there is n input signals (x1, x2, x3,…….xn) , then the neuron performs a linear combination process of all input signals after multiplying each input signal with its associated weight (w_1,w_2, w_3, ……w_n.), the result is given in the following equation (Bataineh, 2012):

$$O_j = \sum_{i=1}^{n} x_i \omega_{ij} \quad (3.1)$$

In order to generate the suitable output y for each input set, the activation function which has different equations is applied on the sum of the inputs as the following:

$$y_j = f(O_j) = f(\sum_{i=1}^{n} x_i \omega_{ij}) \quad (3.2)$$

3.4.3 Multi-layer perceptron

A multi-layer perceptron (MLP) is the other type of ANNs which is formed by one or more hidden layers, in this type the perceptron can also learn non – linear functions, whereas in the SLP the perceptron can only learn linear functions. However, this type of ANNs is more useful than single layer perceptons for practical applications today.

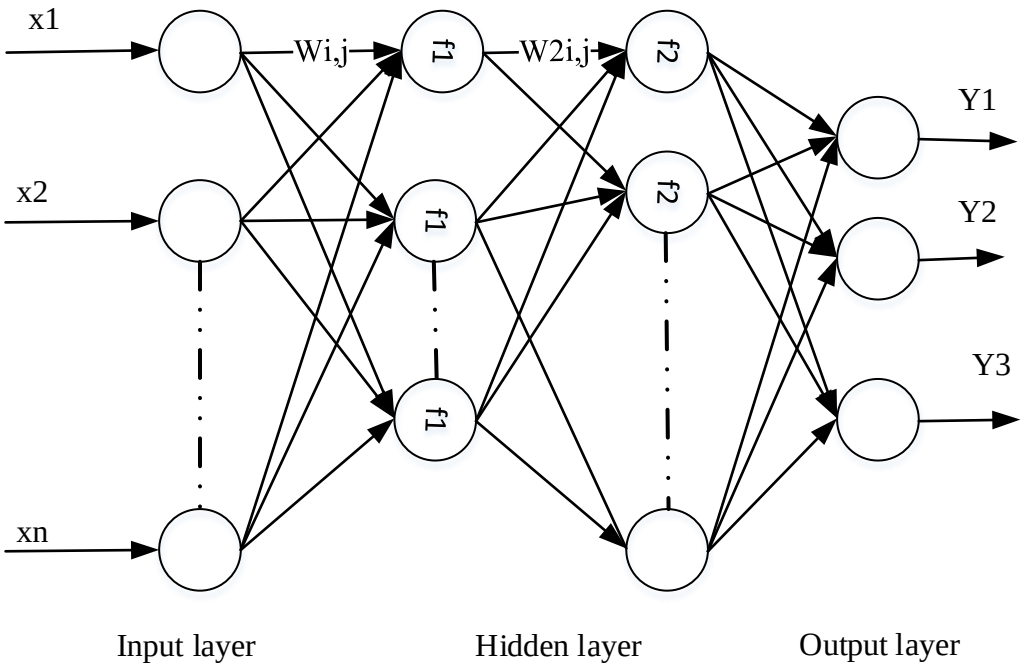


Figure 3.4: Structure of multi-layer perceptron (MLP)

As the Figure 2.4 shows that in MLP the neurons are connected in parallel way in the same layer while the layers are connected in series form. In this example there is one hidden layer consisting of j neurons and f1 is used as an activation function for the outputs of hidden layer, while f2 activation function is used as the function of the final outputs in networks. The types of the activation function will be discussed later in this chapter.

There are different learning methods used in order to train the MLP network, and the back propagation algorithm can be considered as one of the most famous algorithms in today applications.

3.4.4 The synaptic weights

In artificial neural networks the weights can be defined as an adaptive factor which are associated with signals and reflect the importance of the signal in the effect in the output of the system. In other words, whenever the lower the value of weight allocated and associated with input signals, the less importance of that signal impact in the result of the output network. Moreover, in order to adjust the value of weights, various learning methods are used.

3.5 Neural Network Activation Function

Activation function or transfer function is one of the most essential parameter in ANNs. However, In artificial neural networks applying the activation function is the process that follows the input summing process and act as decision making units in ANNs, the primary purpose of applying activation function is to determine if the neuron is activated or not, in addition to its ability to squash or reduce the output of the neural network depending on the used type of activation function.

In this section, we will explain the most popular and commonly used function in the field of artificial neural networks.

3.5.1 Step function

Step function is considered as the simplest type of activation function that is commonly used in binary classification studies especially in single layer perceptrons. This activation function is also called binary step function since its output is 0 or 1. In other words, the input signals are accumulated in the neuron, and if the strength or the value of the resulting

signal is above a certain threshold, the neuron passes the signal .Otherwise; the signal is killed by the neuron and is not propagated further (G. Anderson & McNeill, 1992).

The following equation expresses the fundamental idea of this activation function while the curve chart is shown in Figure 2.5.

$$f(x) = \begin{cases} 0, if\ x < threshold \\ 1, if\ x > threshold \end{cases} \quad (3.3)$$

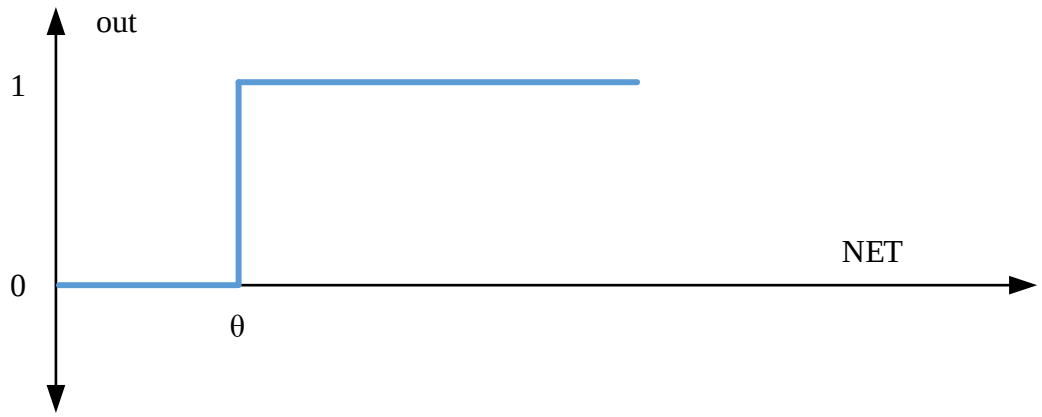


Figure 3.5: The curve chart of step activation function

3.5.2 Linear function

In this type the output of the weighted sum of the inputs is a linear function of its input; with limited output within some band in order to avoid divergence as it shown in Figure 2.6. However, this linear activation function continuous and can have infinite number of outputs. The function of this type is defined by:

$$f(x) = ax \quad (3.4)$$

Where; the value “a” is the slope of the function that controls the output.

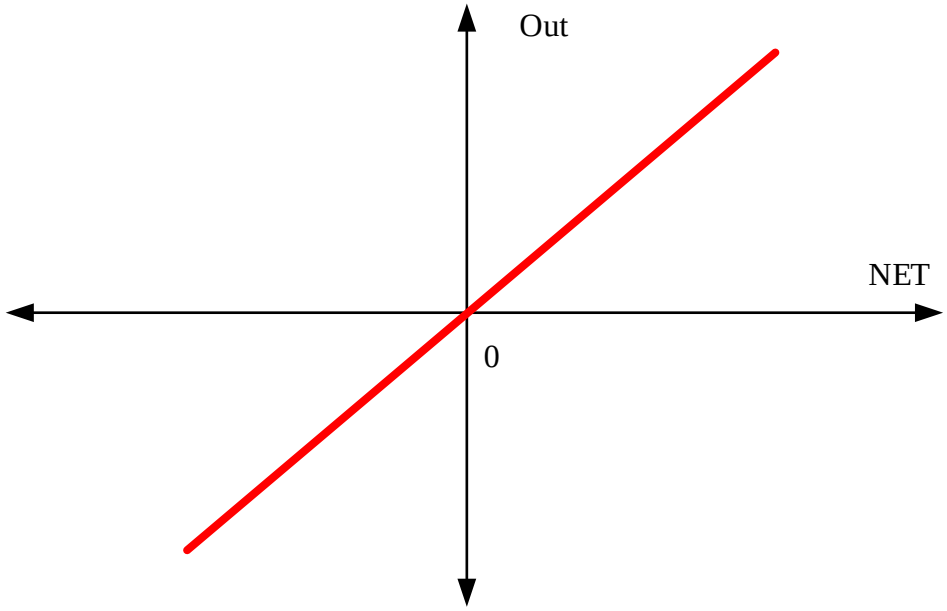


Figure 3.6: The curve chart of linear activation function

3.5.3 Sigmoid function

In the previous section the linear activation function was explained and it was clear that linear function passes the input signals without any significant change, since artificial neural networks perform complex tasks; they need nonlinear activation function in order to make a non-linear decision especially in classifications tasks. One of the most common non-linear functions is sigmoid activation function which is squashes the real-valued number into a range between 0 and 1as it shown in Figure 2.7. Moreover, sigmoid activation function is derivable and continuous in the period of the inputs and it is represented mathematically as (G. Anderson & McNeill, 1992):

$$f(x) = \frac{1}{1+ e^{-x}} \qquad (3.5)$$

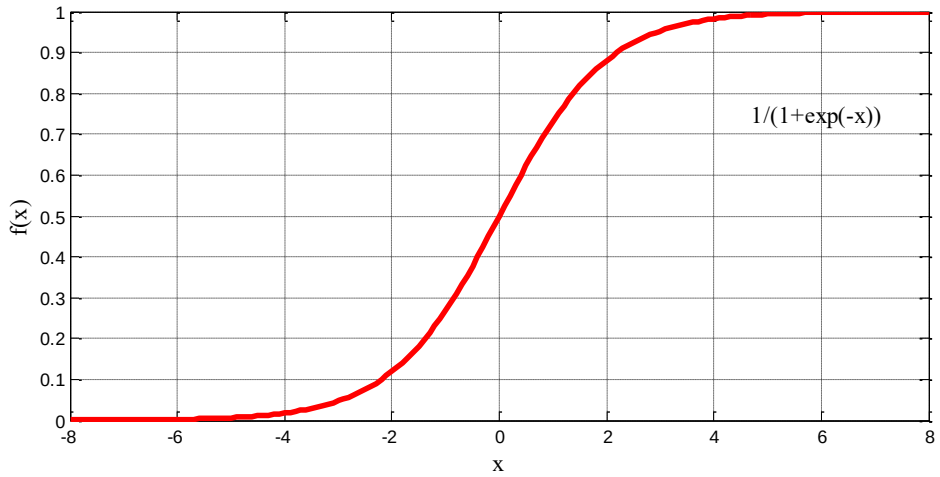


Figure 3.7: The curve chart of sigmoid activation function

The main drawback of this type of activation function is that sigmoid function can cause a neural network to get "stuck" during training. This problem occurs because the output of the sigmoid function is close to zero in case of strongly-negative input.

3.5.4 The hyperbolic tangent activation function

This type of activation function can be considered as two sigmoid functions together, since the hyperbolic tangent activation function squashes the real-valued number into a range between -1 and 1as it shown in Figure2.8, this characteristic is featured by the hyperbolic tangent activation function can addresses the zero-centered problem in sigmoid activation function. The following formula expresses this type mathematically:

$$f(x) = \frac{2}{1+ e^{-2x}} - 1 \qquad (3.6)$$

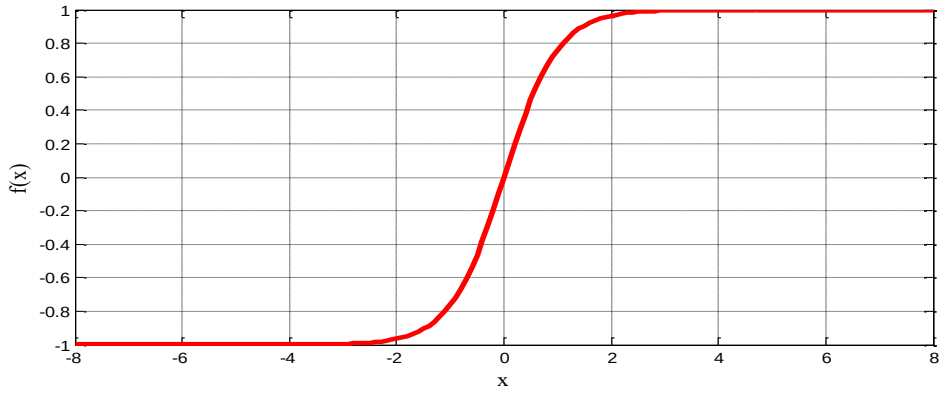


Figure 3.8: The curve chart of the hyperbolic tangent activation function activation function

3.6 Artificial Neural Network Learning and Adaptation

Generally, the term of learning refers to the gaining knowledge activity in order to improve the awareness of studied subjects, this activity is possessed by humans, some animals, and artificial intelligence systems, while the term of learning in the field of machine learning refers to the ability to adapt and change in self-manner depending on environment changes and without being explicitly programmed (Clabaugh, Myszewski, & Pang, 2000).

An artificial neural network is considered as a complex adaptive system which has the ability to change its internal structure in order to adapt with any new environment or parameters. However, this task is performed by adjusting the weights values which are associated with signals to generate the desired output if the networks fed with a given input (Cios & Shields, 1997).

3.7 Types of Learning in ANNs

Basically, learning methods can be categorized into basic methods: supervised learning, unsupervised learning and reinforcement learning which will be explained in the next sections.

3.7.1 Supervised learning

Supervised learning method is considered as dependent learning process, since it takes place under the supervision of a teacher. In this method the artificial neural networks are

learned by providing them with input vector in order to generate its output vector, where each example is consisting of a pair of input vector and its desired output value, for further clarification, during this training phase, the used learning algorithm will help the neural networks in adjusting the weights values in each cycle of training, and this is done through the process of producing an error signal which represents the difference between the desired output and the actual output as it is shown in the Figure 2.9.

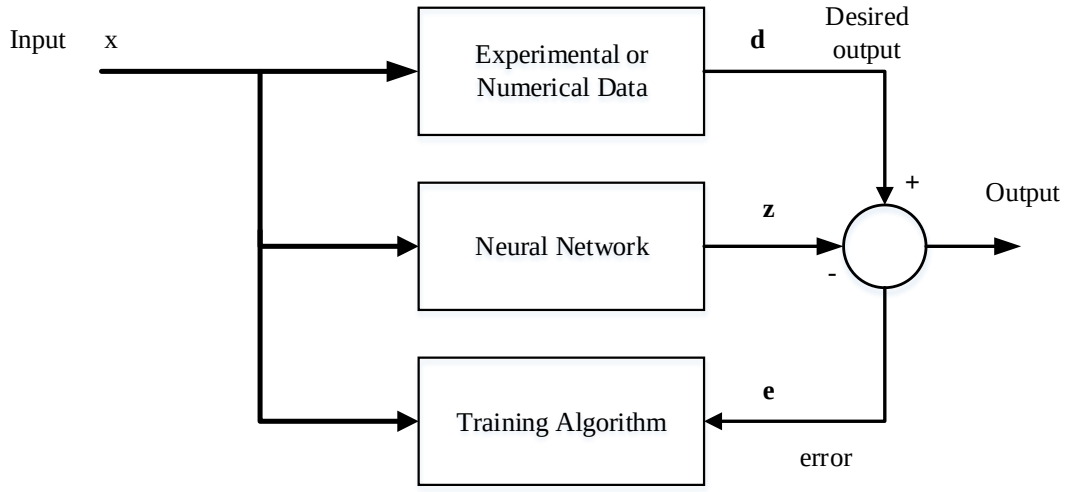


Figure 3.9: Block diagram of supervised learning.

3.7.2 Unsupervised learning

Unsupervised learning method is considered as independent learning process, since it is learning without the supervision of a teacher. However, in this type of learning there is no feedback from the environment as the error signals, that is mean the neural network is required to learn itself depending on some features in the input data in order to adjusting their weights. The figure bellow illustrates the simplest block diagram of unsupervised learning.

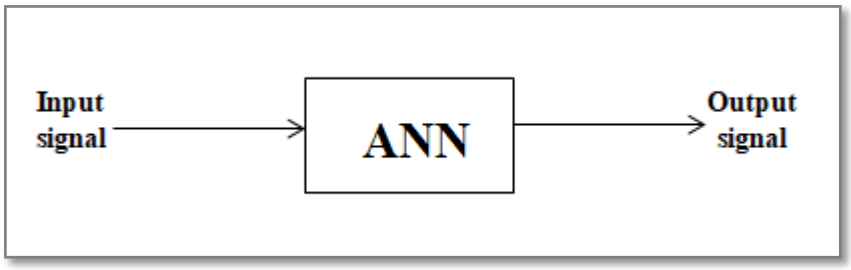


Figure 3.10: Block diagram of supervised learning.

3.7.3 Reinforcement learning

Reinforcement learning and supervised learning methods are similar in that some feedback is given during learning phase, however, in reinforcement learning a target output will be provided instead of the desired output.

This method in learning state that in order to increase the likelihood of the same response of the networks in the right direction, the reward is given based on how well the system performed. In other words, this approach for learning machine bases on affirmation learning feedback; that evaluates the learner's performance without providing standards of correctness in the form of behavioral targets.

3.8 Learning Rules in ANNs

Learning is one of the most important and fundamental characteristics of an artificial neural network that give network their importance in many fields and applications. Basically, this task is performed by adjusting the weights values in order to change the input/output behavior. However, the methods and algorithms used to perform that task are called learning rules.

The most important rules in ANNs learning will be explained in the following sections and the basic ideas will be clarified by equations as well.

3.8.1 Hebbian learning rule

This type of learning rule which is called Hebbian learning rule is considered as one of the oldest learning algorithms, which is simulating the dynamics part in biological nervous. This type of learning rule determines how to alter the weights between neurons.

In 1949 Donald Hebb claimed that if two neurons on either side of a synapse connection are activated simultaneously, then the strength of that synapse between these two neurons will increase selectively, similarity, in ANNs the weight is increased with high correlation between two sequential neurons. However, this rule is adjusting the weights by the following formula which is describing the increment by which the weight of connection increases at time step t (Colin, 1996):

$$\Delta\omega_{ij} = \alpha x_i(t) y_i(t) \qquad (3.7)$$

Where $x_i(t)$ represents the value of pre-synaptic input at time step t while $y_i(t)$ is the value of pre-synaptic output at same time step t; and α is a control variable that control the size of variation in the weights.

3.8.2 Perceptron learning rule

In machine learning this learning rule is considered as supervised learning algorithm of single layer feed-forward networks with linear activation function, which is invented in 1957 by Rosenblatt.

This algorithm is used in case if the problem is linearly separable, in the beginning, the training patterns is provided to the network as inputs, then the output is calculated. In this method Random small values of weights and threshold will be specified in order to modify the weights according to the following formula:

$$\omega_{i+1} = \omega_i + dx, if\ output\ \neq disired\ output \qquad (3.8)$$

Otherwise there will be no weight adjustment, ω_{i+1} is the new weight value while ω_i is the old weight value. In order to compute the output y the activation function will be applied over that net input which can be expressed as follows (Colin, 1996):

$$f(x) = \begin{cases} 0, if\ x > threshold \\ 1, if\ x < threshold \end{cases} \quad (3.9)$$

3.8.3 Delta learning rule (Widrow-Hoff Rule)

In 1960 Widrow and Hoff developed one of the most common learning rules in machine learning field which is called delta learning rule. This learning rule can be considered as supervised learning method with continuous activation function, However delta rule is a gradient descent learning rule that reducing the error for each pattern by minimizing the difference between the net input to the output unit and the target value in order to update the weights. Mathematical the updating of synaptic weight can be done by using in the following equation (G. Anderson & McNeill, 1992; Clabaugh et al., 2000):

$$\Delta\omega_i = \alpha x_i (d - y) \quad (3.10)$$

Where $\Delta\omega_i$ expresses the change in weight for ^{i}th pattern, α is the learning rate, xi is the input value from pre-synaptic neuron, d is the desired output while y represents the actual output.

In the case of there is a difference between the desired and actual output, then updating of synaptic weight is given by:

$$\omega_{i+1} = \omega_i + \Delta\omega_i \quad (3.11)$$

Otherwise, there will be no weight adjustment.

3.8.4 Back propagation algorithm

The back propagation algorithm is a learning algorithm in ANNs which uses the gradient descent method with respect to weights in order to minimize the error function and modifying the weights.

In machine learning field, the back propagation algorithm based on Hebb learning rule can be considered as supervised learning. Moreover, this learning algorithm was named because of its method in updating the values of weights where the error is propagated between layers of the network backwards. Basically, in the training phase the training pairs which are consist of the input and its correspondent desired output are provided to the networks in forward iteration. After the activation function is applied, the actual output is calculated and the comparison process is performed between the desired output and the actual output in order to calculate the error.

Multi-layer perceptron networks can be trained by back propagation algorithm in order to solve several types of problems that include classification, function estimation, and time-series prediction.

3.9 Mathematical Representation of Back Propagation Algorithm

The back-propagation algorithm is a learning algorithm of multilayer feed forward neural network which is based on the gradient descent theory in order to calculate least square error and minimize the LMS between the network output values and the target values for those outputs. Figure 2.12 shows the simplest structure three-layer feed-forward back propagation neural network. The input layers consist of p inputs where these income signals (x1, x2, ……., xp) are sent without any change to the hidden layer as I_n which represent the vector of all inputs. Mathematically, in the hidden layer, the sum of multiplied inputs is calculated as the following (Colin, 1996):

$$S = \sum I_n \, \omega_n + b_n \qquad (3.12)$$

Where W is the wights matrix and b_n is a vector of bias values. Assuming that f1 is a sigmoid activation function which applied to the output of hidden layer then,the output of the applied activation function is expressed as :

$$f_1 = out(S) = \frac{1}{1+ e^{-ax}} \quad (3.13)$$

The output of the function f is considered as the active output of the concerned neuron. This output is ready to be submitted to the next neuron that collects its input from all the previous neurons. Same formulas are applied to that neuron also. The process continues until reaching the output layer that generates the actual output of the neural network. At this stage, the actual output is compared with the expected result to generate the error signal. The error signal is used to update the weights of the network as follows:

$$E = T - O \quad (3.14)$$

Where; O is the actual output of the network, T is the expected output, and E is the error signal. The error signal is the main part of the learning process in the back-propagation algorithm. The value and direction of the error is used in the update formula of the weights of the neural network layers.

The error function is generated based on the error value of the output. This function is used to generate the new weight value for the next iteration:

$$\omega_{new} = \omega_{old} + \eta E_{old} + \alpha(\delta\omega_{old}) \quad (3.15)$$

Where; η is the learning rate value, α is the momentum factor, and $\delta\omega_{old}$ is the previous variation in the weight value. The second term of the previous equation is the change in the weight value that guarantees the error minimisation. The value of the learning rate controls the speed of the learning of the network. The third term on the right hand of the equation is the effect of momentum factor that is very useful to ensure the continuous decrease of the error signal.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter discusses the practical implementation of the proposed methods and topologies in addition to the results of these methods. All proposed and discussed methods in this work were applied and results were obtained using MATLAB environment.

MATLAB 2015a has a powerful image processing set and artificial neural network toolbox. After the implementation of all methods, obtained results will be presented and discussed in this chapter of this work. The experiments were applied using Sony Vaio core i7 microprocessor with 8G ram card using windows 10.

4.2 Processing of Database Images

Our database is composed of the leaves images of 11 different types of plants. Each one of these plants has 20 different images in jpg format. These images were all collected from internet (Inc, 2016). All database images were obtained in RGB format after being manually processed and treated. The total of 220 database images were all put in separate files in the MATLAB data file preparing for the beginning of their processing.

Images of the plant leaves were all converted to gray scale images and segmented using different segmentation methods prior to the application of the neural networks. The next section will discuss in details the application of ANN with the processed database images.

Figure 4.1: Sample of the used plant leaves

Figure 4.1 presents sample of the used plant leaves in this work. These plant leaves can be classified using the outer shape and specie of their leaves.

4.3 Image Segmentation (Edge Detection)

The process of image recognition implies the intervention of different image processing techniques to increase the efficiency of the system. These methods include the image filtering, extraction of special regions of interest, image size reduction, and normalization of image pixels. Different methods of edge detection were applied in this work to test the effect of each of them on the efficiency of our system. Figure 4.2 shows the results of edge detection using the different edge detection methods.

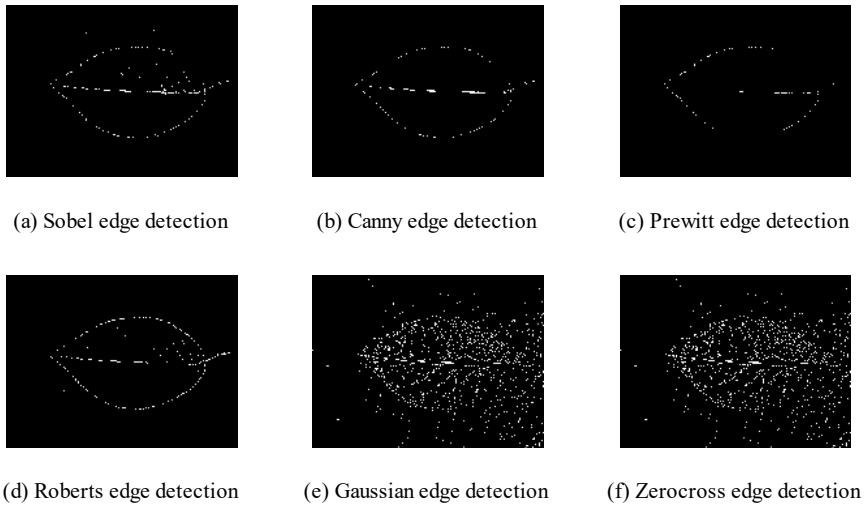
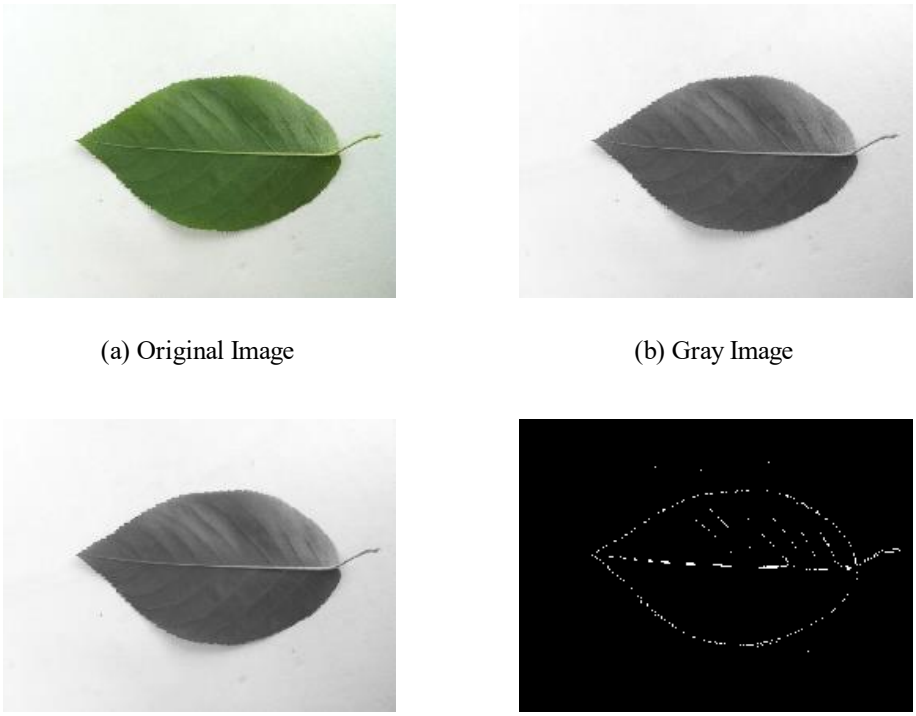


Figure 4.2: Edge detection results

From the figure above, it is noticed that the edge detection using Sobel method has given the best region of interest detection. It will be used as the main edge detection method in this work. The other methods will be used in the purpose of comparing their performance in terms of image detection. The threshold in the different edge detection methods was chosen automatically using the built in MATLAB functions.

4.4 Image Processing Stage

The image processing stage is the first stage in the leaves detection process. This stage includes reading the images in jpg format, converting these images into gray scale format, filtering the image to suppress the noise or any unwanted signal in the image, and finally applying different edge detection processes to identify special areas that contain the main features of the image. Figure 4.3 below presents the different images that result from the application of these different image processing techniques. The first image presents the original image in RGB format. The second image “b” is the gray scale converted image. The image “c” shows the filtered image using Wiener filter. The last part “d” is the segmented version of the leaf showing the edges of the leaf.

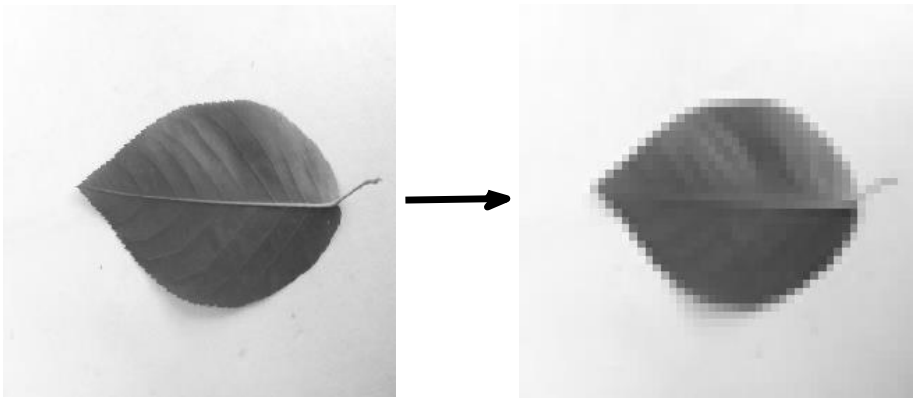


(a) Original Image (b) Gray Image

(c) Wiener Image (d) Segmented Image

Figure 4.3: Image processing stage of the work

After the end of the image segmentation, all database images are going to be resized to small size. This resizing process is very useful in reducing the size of the processed data and the processing time and cost. Generally, images are resized to sizes between 20*20 and 100*100 pixel range. This means that each image can be represented to computer in form of 400-10000 pixels. The choice of the image size is dependent on the training process of the neural network and can differ from application to another. In this work, the image size of 50*50 was chosen as final size for all database images. Figure 4.4 shows the original image and the resized image of the leaf. It is obvious that the resized image has less density and less clear than the original image from the point of human view. However, the computer is less affected by the image density as it deals with pixel values rather than by the scene in the image.



(a) Original size image (b) Resized image 50*50

Figure 4.4: Original image size vs. resized image

After resizing the images to the suitable size, all image pixels are being normalized such that they contain values in the range 0-1. The normalized images are then converted to vectors so that they can be fed to the neural network successively. This is considered the last step in the stage of image processing. In the following part, the structure and details of the used neural network is going to be presented and discussed.

4.5 Artificial Neural Networks Results

The application of the ANN in the recognition process is the last step of the work. This step is composed of two main parts:

- Training of the ANN
- Test of the ANN

The training of the neural network is the process in which the structure of the network is created arbitrarily and then updated continuously to find the best solution that connects the inputs and the outputs successfully.

4.5.1 Structure of the used ANN

The structure of the used artificial neural network is presented in Figure 4.5 below. The network consists of 1 input layer, three hidden layers, and 1 output layer.

Figure 4.5: Structure of the used ANN

Hidden layers are all sigmoid transfer functions while the output layer is a linear transfer function. Table 4.1 presents all the details of the neural network implemented in this work.

Table 4.1: Parameters of the used ANN

Parameter	Value	Parameter	Value
Input size	2500	Learning rate	0.01
Output size	11	Momentum factor	0.1
Hidden size 1	250	MSE	0.0002
Hidden size 2	280	Maximum epochs	28000
Hidden size 3	200		

4.5.2 Training of network using canny edge detection

This network was trained after using the canny edge detection method. The ANN tool of MATLAB is shown in Figure 4.6. The training has stopped after 78 epochs during 16 seconds.

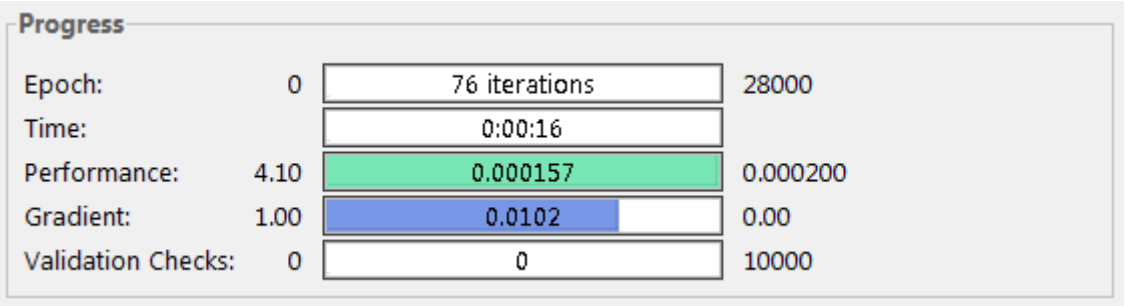


Figure 4.6: ANN tools during the training of network

The training reached MSE of 0.000157. Figure 4.7 presents the curve of the MSE of the training process. It shows that the curve is decreasing fast until it reaches the desired output. The overall performance of this network was 75% during the test and training process.

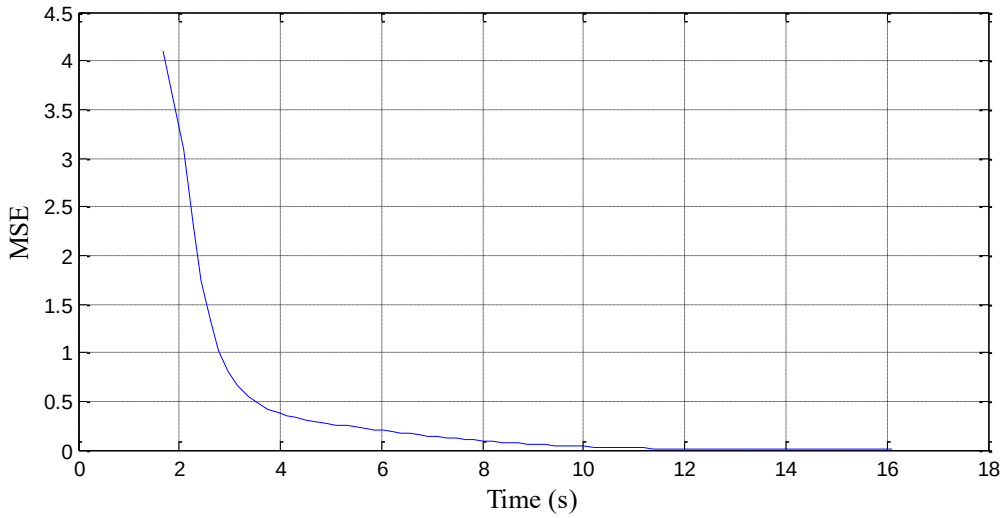


Figure 4.7: MSE curve during the training process

4.5.3 Training of network using sobel edge detection

In this part, the neural network is applied with the images segmented using Sobel edge detection. The parameters of the network are the same as shown in Table 4.1. The overall performance of the network reached 89.5% in this experiment. The training details are depicted in Figure 4.8 below. The training took 1613 iterations to reach MSE of 0.00019 during 4 minutes and 36 seconds.

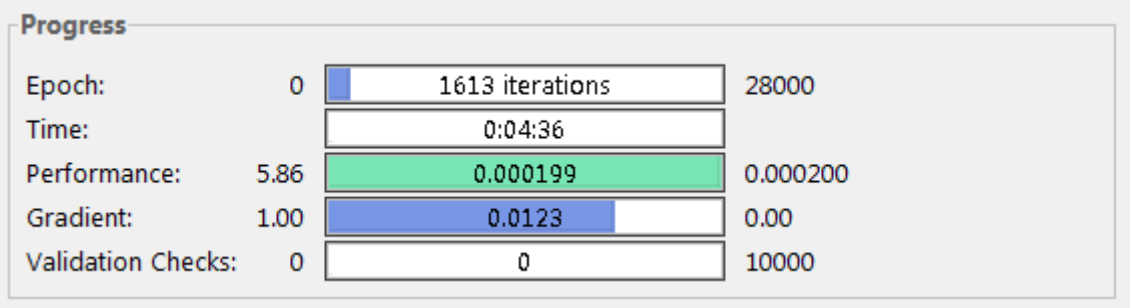


Figure 4.8: Training details of the ANN, Sobel edge method

Figure 4.9 presents the training MSE curve of this network. The curve is showing continuous decrease in the MSE with the training development. Based on the results, it was found that 197 images out of the 220 image were correctly identified in this experiment.

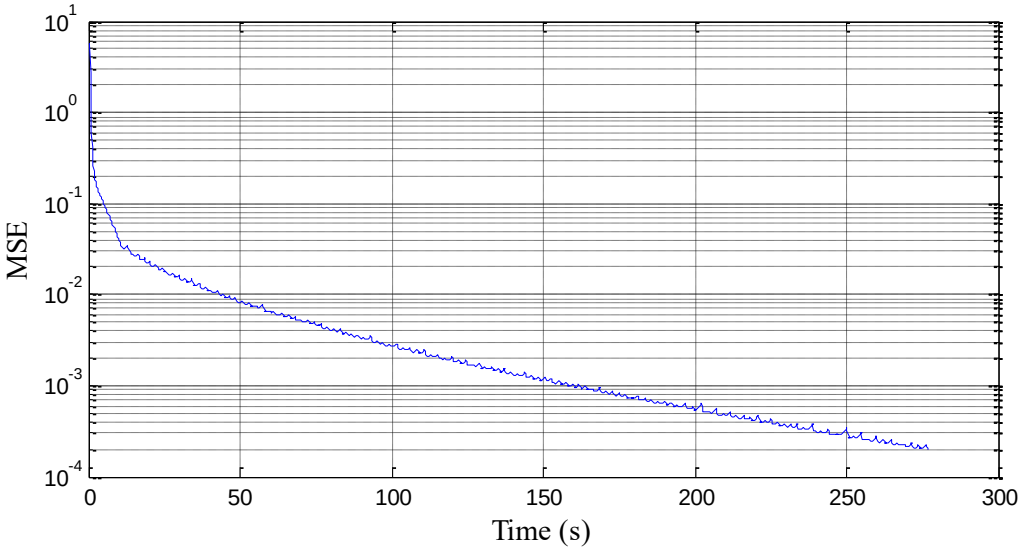


Figure 4.9: MSE curve during the training of the network, Sobel

4.6 Comparison of the ANN Results Using Different Parameters

The results obtained by neural network are subject to change at every experiment dependent on the experiment conditions. Any change in the parameters of the ANN can change the obtained results and affect the overall performance of any neural network application. The training of the neural network is a search for best combination of parameters that can give the highest performance out of the applied network. In this part, the parameters of the network are going to be varied one at a time to test the performance of the network.

4.6.1 Varying the momentum factor

The parameters of the neural network will be kept unchanged and the momentum factor was varied to find the performance of the network.

Table 4.2: Performance change in function of Momentum factor

Momentum value	Performance	Momentum value	Performance
0.1	87.3%	0.05	89%
0.01	91%	0.02	90%
0.001	85%	0.008	92%

4.6.2 Varying the learning rate

Learning rate is varied in this experiment while all other parameters where kept constant. The momentum factor is fixed to 0.008. The hidden layer sizes are 200, 280, and 280 neurons. Table 4.3 presents the obtained results using different learning rate values. It was found that the learning rate of 0.09 has given best performance. This value will be used in the next experiments.

Table 4.3: Performance of ANN in function of the learning rate

Learning rate	performance	Learning rate	performance
0.001	90%	0.05	91.8%
0.08	93.6%	0.01	88%
0.1	93.2%	0.09	94%
0.5	93.5%	0.002	91%

4.6.3 Varying the hidden layers sizes

Hidden layers are very important in the determination of the behavior of the neural network during the training and the test processes. Choice of the size of the hidden layer is an accurate process because the larger the size of hidden layers the slower the program is different combinations of hidden layer sizes were experimented to test the performance of the network. Table 4.4 shows the performance results with different layer sizes.

Table 4.4: Performance of network in function of hidden layer size

Layer sizes	performance	Time (s)	iterations
[200 , 280 , 280]	94%	185	2250
[20, 280, 280]	91.2%	120	2500
[20, 20, 280]	85%	124	3878
[100, 100, 180]	90.5%	136	1653
[300, 300, 300]	93.6%	366	1587
[300, 100, 300]	92.3%	397	1832

After finding the best values for the parameters of the neural network, the network is going to be trained based on these parameters.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

The plant leaves identification is an important process in different fields from medical plants collection and classification to the study of the environmental life. The use of computerized systems for the identification of different types of plants can reduce the efforts done by scientists and experts in the environmental sciences. Some plants are very rare and can't be identified without the presence of huge books and experienced experts. Neural networks can be implemented to identify the different plants with high performance and less effort. Neural networks can store huge data about these plants and restore them whenever they are needed.

Back propagation artificial neural network algorithm has gained a great importance since the last two decades. It has proven great capability to perform well in different fields like image processing, security, data analysis, and forecasting weather and products prices. They are working in similar way the human brain works. Neural networks are consisted of interconnected processing units called neurons. Information is passed between these neurons in a manner that gives them the ability to generate results and learn patterns of data.

This work is concerned by the study of the application of neural network in the identification of plants specimen. The application is carried out using the back propagation algorithm to classify 220 images of 11 different plants. Different image processing techniques were used to simplify the processing of the images using ANN. Image types conversion, image filtering, and image segmentation were all used in this work. The implementation of this classification system using neural network was applied and tested using different parameters.

The image segmentation using different segmentation methods was discussed in this work. Results of segmentation have shown that Sobel segmentation method gives the best performance in terms of finding the region of interest in leaves images. Canny edge

detection has also shown good performance in finding the edges of the leaf; however Sobel method was chosen to be used as the main segmentation method.

Parameters of the back propagation and neural network have been proven to have an important effect on the performance of the neural network. The modification of learning rate and momentum factor has increased the efficiency of the network in the plant classification.

Obtained results have shown that the back propagation algorithm and the neural network is capable to classify the leaves images with high performance and minimum error. The proposed system can be extended for larger amount of data and different types of plants. As a future work, it is recommended to use natural database from the field and to extend the research to include more classification criterions such that its performance can be ensured to be higher.

REFERENCES

Amin, A. H. M., & Khan, A. I. (2013). One-shot Classification of 2-D Leaf Shapes Using Distributed Hierarchical Graph Neuron (DHGN) Scheme with k-NN Classifier. *Procedia Computer Science*, *24*, 84–96. https://doi.org/10.1016/j.procs.2013.10.030

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.

Boran Sekeroglu. (2004). *Intelligent Banknote Identification System (IBIS)*. Near East University.

Chaple, G. N., Daruwala, R. D., & Gofane, M. S. (2015). Comparisions of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA. In *2015 International Conference on Technologies for Sustainable Development (ICTSD)* (pp. 1–4). IEEE. https://doi.org/10.1109/ICTSD.2015.7095920

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

Clabaugh, C., Myszewski, D., & Pang, J. (2000). Neural Networks. Retrieved May 1, 2017, from http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/index.html

Colin, F. (1996). *Artificial Neural Networks* (1.1). University of Paisly.

Dongare, A., Kharde, D. & Kachare, A. (2012). Introduction to Artificial Neural Network. *International Journal of Engineering and Innovative Technology*, *2*(1), 2277–3754.

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

Inc, K. (2016). Leaf Classification. Retrieved January 20, 2018, from https://www.kaggle.com/c/leaf-classification

Kadir, A., Nugroho, L. E., Susanto, A., & Santosa, P. I. (2013). Leaf Classification Using Shape, Color, and Texture Features. Retrieved from http://arxiv.org/abs/1401.4447

Khajwaniya, K. K., & Tiwari, V. (2015). Satellite image denoising using Weiner filter with SPEA2 algorithm. In *9th International Conference on on Intelligent Systems and Control* (pp. 1–6). India. https://doi.org/10.1109/ISCO.2015.7282324

Mallah, C., Cope, J., & Orwell, J. (2013). Plant Leaf Classification using Probabilistic Integration of Shape, Texture and Margin Features. In *Computer Graphics and Imaging / 798: Signal Processing, Pattern Recognition and Applications*. Calgary,AB,Canada: ACTAPRESS. https://doi.org/10.2316/P.2013.798-098

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

Mohan, R., Mridula, S., & Mohanan, P. (2016). Speckle Noise Reduction in Images using Wiener Filtering and Adaptive Wavelet Thresholding. *IEEE*, 2860–2863. https://doi.org/10.1109/TENCON.2016.7848566

Robert, F., Simon, P., Ashley Walker;, & Wolfart, E. (2003). Hypermedia Image Processing Reference. Retrieved from http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

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.

Shen, D., & Tang, Z. (n.d.). Canny Edge Detection Algorithm for Image Processing Using FPGA. In *Second International Conference on Electric Information and Control Engineering* (pp. 419–442). Washington. https://doi.org/10.1109/ICEICE.2012.110

Wang, J., Peng, Y., Wang, X., & Peng, Y. (2015). Study on algorithm of image restoration based on Stochastic Resonance and Weiner filtering. In *5th International Conference on Electronics Information and Emergency Communication* (pp. 244–247).

Zollitch, C. (2016). Grey scale image. Retrieved April 2, 2017, from http://www.stemmer-imaging.co.uk/en/knowledge-base/grey-level-grey-value/

APPENDIX

LIST OF PROGRAM

```
clc
clear all
close all
folders = {'a','b','c','d','e','f','g','h','i','j','k'};
progpath = cd;
datapath = strcat(progpath,'\data');
% if(exist('edge','dir')),  rmdir('edge'); end
% if(exist('smallsize','dir')),  rmdir('smallsize'); end
% if(exist('wiener','dir')),  rmdir('wiener'); end
% if(exist('gray','dir')),  rmdir('gray'); end
%cd(datapath);
clc
cont = {'gray' , 'wiener' , 'smallsize' , 'edge' , 'canny' ,
'prewitt' , 'roberts' , 'log' , 'zerocross'};
for i=1:length(cont)
    if(exist(char(cont(i)),'dir')==7)
        rmdir(char(cont(i)) , 's');
    end
end

for i=1:length(cont)
    mkdir(char(cont(i)));
    cd(char(cont(i)));
    for k=1:length(folders),
        clc; mkdir(folders{k});
        clc;
    end
    cd(progpath);
end


for i = 1:length(folders)
    for j = 1:20


        imagepath{i, j} =
strcat(progpath,'\data','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        graypath{i, j} =
strcat(progpath,'\gray','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
```

```
        wienerpath{i, j} =
strcat(progpath,'\wiener','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edgepath{i, j} =
strcat(progpath,'\edge','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edge1path{i, j} =
strcat(progpath,'\canny','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edge2path{i, j} =
strcat(progpath,'\prewitt','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edge3path{i, j} =
strcat(progpath,'\roberts','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edge4path{i, j} =
strcat(progpath,'\log','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');
        edge5path{i, j} =
strcat(progpath,'\zerocross','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');


        smallsizepath{i, j} =
strcat(progpath,'\smallsize','\',folders{i},'\',folders{i},'
(',num2str(j),').jpg');

    end
end


[iend jend] = size(imagepath);
count=0;
generalInput  = zeros(2500, jend*iend);
Tmat = eye(11);
generalOutput = zeros(iend, jend);
for j=1:jend
    for i=1:iend
        a = imread(imagepath{i, j});
        b = rgb2gray(a);
        c = wiener2(b,[3 3]);
        d = edge(c,'sobel');
        ed1 = edge(c,'canny');
        ed2 = edge(c,'prewitt',0.2);
        ed3 = edge(c,'roberts',0.2);
        ed4 = edge(c,'log',0.2); % L^place of gaussian
        ed5 = edge(c,'zerocross');


        x = figure(1) ;
        set(x,'OuterPosition',[50 50 650 650])
```

```
        subplot(231); imshow(d);
xlabel('Sobel','fontsize',12,'fontname','times');
        subplot(232); imshow(ed1);
xlabel('Canny','fontsize',12,'fontname','times');
        subplot(233); imshow(ed2);
xlabel('prewitt','fontsize',12,'fontname','times');
        subplot(234); imshow(ed3);
xlabel('roberts','fontsize',12,'fontname','times');
        subplot(235); imshow(ed4); xlabel('Logarithm of
Gaussian','fontsize',12,'fontname','times');
        subplot(236); imshow(ed5);
xlabel('zerocross','fontsize',12,'fontname','times');


        e = imresize(c,[50 50]);
        ee = imresize(d,[50 50]);
        f  = reshape(e,1,2500);
        ff = reshape(ee,1,2500);
        y = figure(2);
        set(y,'OuterPosition',[720 50 650 650])
        subplot(221); imshow(a); xlabel('Original
Image','fontsize',12,'fontname','times');
        subplot(222); imshow(b); xlabel('Gray
Image','fontsize',12,'fontname','times');
        subplot(223); imshow(c); xlabel('Wiener
Image','fontsize',12,'fontname','times');
        subplot(224); imshow(d); xlabel('Segmented
Image','fontsize',12,'fontname','times');
        imwrite(b,graypath{i,j},'jpg');
        imwrite(c,wienerpath{i,j},'jpg');
        imwrite(d,edgepath{i,j},'jpg');
        imwrite(ed1,edge1path{i,j},'jpg');
        imwrite(ed2,edge2path{i,j},'jpg');
        imwrite(ed3,edge3path{i,j},'jpg');
        imwrite(ed4,edge4path{i,j},'jpg');
        imwrite(ed5,edge5path{i,j},'jpg');



        imwrite(e,smallsizepath{i,j},'jpg');
        count = count+1;
        perc = 100*count/(iend*jend);
        clc
        fprintf('                        %2.0f%% finished\n',perc);
        generalInput(:,count) = f ;
        generalOutput(:,count) = Tmat(:,i);
        pause(0.1);
    end
end

cd(progpath);
```

```
end
    clc
momentum = 0.1;
learning_rate = 0.01;
transfer_functions = {'tansig','tansig','logsig'};
layer_size = [250 280 200];
g = newff(generalInput,generalOutput,layer_size,
transfer_functions ,'traingdx');
nets = init(g);
nets.trainParam.lr = learning_rate;
nets.trainParam.show = 1000; % Frequency of progress displays (in
epochs).
nets.trainParam.epochs = 100000; % Maximum number of epochs to
train.
nets.trainParam.mc = momentum; % Momentum Factor.( for change
*************
nets.trainParam.min_grad = 0;
nets.trainParam.max_fail = 10000 ;
nets.trainParam.mu_max=1e20;
nets.divideParam.valRatio=0;
nets.divideParam.trainRatio=0.6;
nets.divideParam.testRatio=0.4;
nets.performFcn ='mse';
nets.trainParam.goal = 0.0002;
nets.trainParam.epochs = 28000;
[netR,TR] = train(nets,generalInput,generalOutput);
out = sim(netR,generalInput);
out = out>0.3;
count = 0 ; co = 0;
for i=1:20
    for j=1:size(out,1)
        count = count+1;
        if(out(j,count) == 1)
            co = co+1;
        end
    end
end
fprintf('                  Overall performance is : %2.1f%%
\n',100*co/220);
```