

**LEAVES RECOGNITION SYSTEMS USING
ARTIFICIAL NEURAL NETWORK**

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

**by
YÜCEL İNAN**

**In Partial Fulfillment of the Requirements for the
Degree of Master of Science
in
Computer Engineering**

NICOSIA 2015

**Yücel İnan: LEAVES RECOGNITION AND CLASSIFICATION USING
ARTIFICIAL NEURAL NETWORK**

Approval of Director of Graduate School of Applied Sciences

Prof. Dr. İlkey SALİHOĞLU

We certify that this thesis is satisfactory for the award of the degree of

Master of Science in Computer Engineering

Examining Committee in Charge:

DECLARATION

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: Yücel İnan

Signature:

Date:

ACKNOWLEDGEMENTS

I thank God Almighty for this master's thesis, which without His blessings would not be possible for me to complete. I would like to express my deepest gratitude to my adviser, Assist. Prof. Dr Boran Şekeroğlu, for his patience and for his guidance. I also would like to thank everyone who helped me in completing this work.

ÖZET

Bitki tanıma, biyolojik ve tıp bilim alanlarında önemli bir rol oynar. Şifali bitkiler yüksek bir kesinlikle tanınıp sınıflandırılmalıdırlar. Sınıflandırmada meydana gelen bir hata kötü sonuçlar doğurabilir. Örneğin, zehirli bir bitkinin yaprağı, şifa nitelikli bir bitki ile karıştırılırsa, ölümcül sonuçlar meydana gelebilir. Yada şifa özelliği olmayan bir bitkinin yaprağı önemli şifa nitelikleri olan bir bitki ile karıştırılabilir. Bu nedenle doğru sınıflandırma çok önemlidir. Normal şartlarda bitki sınıflandırılması elle ve tecrübeye dayalı olarak uzman bireyler tarafından yapılır. Fakat bu şekilde sınıflandırma uzmanın tecrübesi ve bilgisi ile sınırlı olur. Bu bakımdan bilgisayar teknolojisinin sunduğu hız ve kesinlik özelliklerinden faydalanıp daha iyi sınıflandırma yapılabilir. Bu fikirden yola çıkılarak, bu çalışma ile yaprak tanıyan yapay sinir ağlarına dayalı bir otomatik bitki tanımlama sistemi önerilmektedir.

Son 50 sene içerisinde, sinir ağları kullanımı bilimde büyük bir devrim olarak görülmüştür. Sinir ağları, yüksek verimlilik ve düşük gider özelliklerinden dolayı, değişik bilimsel uygulamalarda sıkça başvurulan bir yöntemdir. Özellikle biometrik sistemler ve kontrol sistemleri gibi tanıma ve sınıflandırma gerektiren alanlarda sıkça kullanılmaktadırlar. Bitki tanımadaki da tavsiye edilen metodlar arasındadırlar.

Bu çalışma özel olarak, bitki tanıma ve sınıflandırma için, “yaprak tanıyan geri yayılım tipi sinir ağı” üzerinedir. Çalışmada değişik bitkilerin yaprak görüntüleri elde edilip yapay sinir ağı sistemine girdi olarak verildi. Eğitilmiş sinir ağı, eğitim safhası sırasında elde ettiği bilgi sayesinde, değişik yaprak görüntülerini tanıma yetisi elde etti. Bazı eğitim safhalarında görüntülere değişik tipte gürültü katıldı ve sistemin gürültülü görüntü tanıma yeteneği iyileştirilmeye çalışıldı.

Değişik bilimsel çalışmalarda [12-15, 17, 19] alınan iyi sonuçlar, yapay sinir ağlarının yaprak tanımadaki yüksek verimliliğini göstermektedir. Bu çalışmada gürültülü görüntüler üzerine yapılan deneylerde %97 tanıma oranı elde edildi. Gürültüsüz görüntülerle yapılan deneylerde ise %95 tanıma oranı elde edildi.

ABSTRACT

Plant identification is an important field of biological and medical sciences. Medicinal plants must be classified and recognized with high accuracy. Classification errors can lead to high costs and losses. For example if a poisonous plant is classified as medicinal plant, this can lead to fatal cases. Also it can be dangerous, if a plant that has no medicinal property is classified as a medicinal plant. The classification is done normally manually and based on the experience of the human classifier. In this case the classification process is limited by the experience and the knowledge of the human expert. On the other hand exploiting the accuracy and speed of the computer technology can be very useful in creating high performance plant classification system based on the leaf recognition. Based on this idea, this thesis proposes an automatic plant identification process using an artificial neural network that can recognize leaf images of the plants.

The use of neural networks in science has seen a huge revolution in the last decades. Neural networks have been introduced in different scientific applications and proved their high efficiency with minimum costs. It is used mostly in different classification and recognition tasks as like as biometrics and control systems. They are proposed also in the field of plants identification.

This thesis proposes the use of artificial neural network back propagation algorithm for leaf recognition. The images of different plants are acquired and are processed as an input to the artificial neural network. The trained network recognized leaf images based on the information gained during the training process. Noise is added to several image sets in order to test the ability of the artificial neural network in recognizing noisy images. The results obtained from the different scientific researches [12-15, 17, 19] showed the high efficiency of artificial neural networks for leaf recognition. In this thesis two different sets of experiments are designed to assess the performance of the proposed leaf recognition system. In the first set of experiments noise is added to the leaf images. The system achieved 97% recognition rate with the noisy images. In the second set of experiments several different images are used for the same plant leaf, varying in size, orientation and shape. The system achieved 95% recognition rate in the second experiment.

CONTENTS

ACKNOWLEDGEMENT	i
ÖZET	ii
ABSTRACT	iii
CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	viii
1. INTRODUCTION	1
2. ARTIFICIAL INTELLIGENCE	4
2.1 Overview.....	4
2.2 Artificial Intelligence.....	4
2.3 Why Artificial Intelligence.....	4
2.4 The Characteristics of Artificial Intelligence system include.....	5
2.5 Expert System.....	5
2.6 Genetic Algorithm.....	6
2.6.1 Why genetic algorithm.....	6
2.7 Fuzzy Logic.....	7
2.7.1 Applying truth values.....	7
2.8 Neural networks.....	8
2.9 Summary.....	9
3. ARTIFICIAL NEURAL NETWORK	10
3.1 Overview.....	10
3.2 Brief History of Neural Networks.....	10
3.2.1 Neural Network.....	10
3.2.2 Why use Neural Networks.....	11
3.3 Biological Human brain.....	11
3.3.1 From Human Neurons to Artificial Neurons.....	13
3.4 Artificial Neuron Models.....	13
3.4.1 A simple Artificial Neuron.....	14
3.5 Learning in Artificial Neural Networks.....	15
3.6 Learning Types.....	15
3.6.1 Supervised Learning.....	15
3.6.1.1 The perceptron.....	16
3.6.1.2 Back propgation Algorithm.....	16
3.6.1.3 The Hopfield Algorithm.....	16
3.6.2 Unsupervised Learning.....	16
3.6.2.1 Kohonen's self-organizing maps.....	16
3.6.3 Reinforcement Learning.....	17
3.7 Adaptive Network and Systems.....	17
3.7.1 Activation Function Types.....	17

3.7.1.1 Uni-polar Sigmoid Activation Function.....	17
3.7.1.2 Bipolar Sigmoid Function.....	18
3.7.1.3 Hyperbolic Tangent Function.....	18
3.8 Back propagation Artificial Neural Network.....	19
3.9 Summary.....	22
4. ARTIFICIAL NEURAL NETWORKS APPLICATIONS.....	23
4.1 Overview.....	23
4.2 Historical overview of Leaves recognition.....	23
4.3 Why Leaves Recognition.....	24
4.4 Find the leaf venation pattern.....	25
4.5 Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features.....	25
4.6 Disease Detection and Diagnosis on plant using image Processing	28
4.7 Neural Network Application on Foliage plant Identification.....	29
4.8 Early Detection on Pests on Leaves.....	30
4.9 Plant Leaves Diseases detection using Image Processing Techniques.....	31
4.9.1 Plant diseases analysis and its symptoms.....	31
4.10 Summary.....	33
5. LEAVES RECOGNITION and EXPERIMENTAL RESULTS.....	34
5.1 Overview.....	34
5.2 Structure of the System.....	36
5.3 Methodology.....	37
5.4 Database Collection.....	42
5.5 General Experiment.....	42
5.6 Training Process.....	46
5.7 Some results from other researches	49
5.8 Summary.....	49
CONCLUSION.....	50
REFERENCES.....	52
APPENDIX A.....	55
APPENDIX B.....	58

LIST OF FIGURES

Figure 2.1	Expert System.....	6
Figure 2.2	Fuzzy Logic.....	8
Figure 2.3	Basic Artificial Neural Network architectures(Single layer on the left and multi layer on the right).....	8
Figure 3.1	Biological neuron.....	12
Figure 3.2	Architecture of neural networks.....	13
Figure 3.3	A simple neuron.....	14
Figure 3.4	Uni-Polar Sigmoid Function.....	18
Figure 3.5	Bi-Polar Sigmoid Function.....	18
Figure 3.6	Hyperbolic Tangent Function.....	19
Figure 3.7a	A Feed Forward Back-Propagation Network.....	20
Figure 3.7b	Output calculation on the Feed Forward Back-Propagation Network.....	20
Figure 4.1	Leaf Images.....	25
Figure 4.2	Sample images of infected leaves.....	26
Figure 4.3	Detection of infected region for a rose leaf.....	27
Figure 4.4	Infected leaf of cotton plant.....	28
Figure 4.5	Illustration of vein processed by using morphological operation...	29
Figure 4.6	Input image.....	30
Figure 4.7	Gray scale.....	30
Figure 4.8	Bacterial leaf spot.....	32
Figure 4.9	Mosaic virus.....	32
Figure 4.10	Fungal diseases on leaves.....	32
Figure 5.1	Block diagram of the System.....	36
Figure 5.2	Block diagram of the preprocessing phase of the training and test images.....	38
Figure 5.3:	50x50 Training set images for the Alligator pear. The order is from left to right as it explained above. The first image on the left has Gaussian noise with mean 0.02 and variance 0.005.The last one, the right most image is original image.	39

Figure 5.4 50x50 Level I, Test set images for the Alligator pear. The order is from left to right as it explained above. The last one, the rightmost image is with Poisson noise.	39
Figure 5.5 50x50 Level II, Test set images for the Alligator pear. The order is from left to right as it explained above. The last one, the rightmost image is with Poisson noise.	40
Figure 5.6 50x50 Level III Test set images for the Alligator pear. The order is from left to right as it explained above. The last one, the rightmost image is with Poisson noise.	40
Figure 5.7 Sample of the images.....	43
Figure 5.8 Original RGB image and Gray scale image.....	44
Figure 5.9 Resized gray scale image.....	44
Figure 5.10 Curve of the MSE during the training.....	46
Figure 5.11 MATLAB interface of neural networks during training process	47

LIST OF TABLES

Table 4.1	Detected diseased region of various leaves.....	27
Table 5.1	Below are shown noise parameters for testing images. Same Poisson noise added to all levels.....	41
Table 5.2	Training parameters of the ANN used for the first experiment.....	46
Table 5.3	Overall results for the level I, II, III	48
Table 5.4	Some recognition results from different recent researches.....	49

CHAPTER 1

INTRODUCTION

The classification of plants and different types and classes of creatures is one of the most important fields of biology. All creatures are classified into classes and sub-classes based on the similarity and dissimilarity among them. The word “taxonomy” is derived from the Ancient Greek: τάξις which means arrangement or classification; and νομία which means method. So it means the science that is interested in the methods of classification of plants and animals.

The classification of plants is very important in grouping plants into different ranks and classes based on different classifiers or categories. It puts each group of plants having some common properties into classes. Also the classes are then divided into sub-classes and types to differentiate among the elements of the class. This classification is very important to help scientists to study the common behaviors and properties of the plants. Especially those plants used in the medicine or medical plants.

In the past before the invention of digital cameras and computerized systems; people were using their own absolute experience in defining different types of medical plants. The risk of using the wrong plant for medicine extraction increases with the lack of experience and can cause fatal error that can cause the death of some patients. The existence of digital devices and possibilities of computer vision has encouraged the botanists and computer scientists to develop computerized systems or semi automatic systems for plant classification or recognition based on different features. Different researches have treated the problems of the plant classification and mentioned different methods of recognition for these plants. Expert systems were considered also in the plant recognition and classification task.

The last century has seen a very great development in artificial intelligence and machine vision where a lots of pattern recognition and classification tasks were investigated by using automatic computer systems. These applications include finger print recognition which has been developed and face recognition which is developing increasingly.

Artificial neural networks have entered the race of pattern recognition and classification for decades due to their simplicity of implementation and ease of use;

in addition to their flexibility for different applications and the high efficiency that they achieved. The development of digital processors has also encouraged the use of neural networks in many sciences. Neural networks use mathematical equations to imitate the structural construction and functional principle of the biological brain. Neural networks implement a structure similar to human brain to learn the pattern among different elements and apply themselves based on the acquired knowledge. This knowledge is then applicable on other elements that may have the same pattern or not.

Neural network systems gain their knowledge and develop their experience over the time by using examples to reinforce the weights of connections between their neurons. They use the example and error in repetitive check and adaptation or rearrangement of themselves to suit the system they are trying to describe. This repetitive task is called training or learning of neural networks. Whenever the networks develop a correct relation between their input and output examples then they are called to be trained.

This thesis concentrates on the employment of ANN system to classify different medical plants and differentiate among them using their leaves. Different leaves collected arbitrary from different medical plants and their images were taken using a digital camera. The images were then processed and used in the training of a neural network computer system. Results and methods are described and discussed in the thesis [1]. The aim of this thesis is the development of artificial neural networks for recognition of leaves. Thesis includes five chapters and conclusion. Chapter 1 presents a brief introduction to Artificial Intelligence and also describes fuzzy logic, expert systems, genetic algorithms and neural networks. Chapter 2, explains briefly Artificial Neural Networks. It describes the history and the back propagation learning algorithm. Chapter 3 presents a brief historical overview on leaves recognition and mentions different methods on the foliage plant identification systems, which incorporate shape, vein, color, and texture features for classification. Chapter 4, includes the methodology of the research and the different steps of it; starting by collecting the data base and ending with the testing stage. The implementation of the noise and processing of the images is an important stage in the recognition procedure. The processing of the images is done by using MATLAB which includes adding the noise to the images in addition to resizing and changing the type of images to reduce processing expenses. The pre-processing stage is a very important

stage for a prosperous recognition rule. The choice of image size is important as it affects directly the results of the program in addition to the running time.

CHAPTER 2

ARTIFICIAL INTELLIGENCE

2.1 Overview

This chapter presents a brief introduction to Artificial Intelligence and also describes fuzzy logic, expert systems, genetic algorithms and neural networks.

2.2 Artificial Intelligence

Artificial Intelligence is one of the branches of the Computer Science that tries to develop systems that can think like human beings to solve complex problems. The intelligence that is the ability to reason in the system is “artificial” that is designed by man. In general Artificial Intelligence has strong ties with other branches of Science like *Philosophy, Biology, Maths, Psychology, and Cognition* [2].

2.3 Why Artificial Intelligence?

Computers are suitable to carry out computations, using unchangeable programmed principles. This property permits artificial systems to operate efficiently and correctly in high speed over and over again, whereas for human beings doing so is not easy. But on the other hand contrary to human intelligence, computers have difficulties in knowing exact conditions and conforming to new conditions. Artificial Intelligence helps to make better machine behavior in tackling complex tasks [2].

2.4 The characteristics of Artificial Intelligence System

Data and information employed to grow computer systems that show features of intelligent behavior.

Intelligent behavior has numbers of various parts to it, containing the capacity to:

- Learning from trial and implementation, which is the information that is gathered from those trials
- handling complicated conditions
- solving problems where time and data lost are essential
- calculating what is worth
- moving quickly and adapt to new conditions
- understanding and perceiving
- being methodological and operating together
- being inventive and creative

These things as like people are natural better at since that is how human brains work. The basic computer is better at performing easy duties at very high speed. So that does creating a computer system with intelligent behavior [2].

2.5 Expert System

The Expert System is basically a computer program that has inside a *knowledge base* and a set of *algorithms*. An expert system is an artificial intelligence implementation that employs a knowledge basis of people knowledge to help to solve troubles.. The extent of trouble finding a solution is based on degree of the data and systems prevailed from the people expert. Expert systems are patterned to carry out at a people expert degree In repeated performance will carry out they good lower , good over that of a separate expert. The expert system have as origin its replies to start by running the knowledge basis to the end of an *inference engine*, a software programming that moves with someone and methods the ending from the systems and data in the knowledge base. The figure 2.1 shows the general structure of an expert system that was explained above [5].

Expert systems are employed in implementation like as medical diagnosing, machines reparation, investing analysis, financial, housing and indemnify planning, way programming for deliverance vehicles, make an agreement wishes, advising for self-service clients, creation limitation and learning [4].

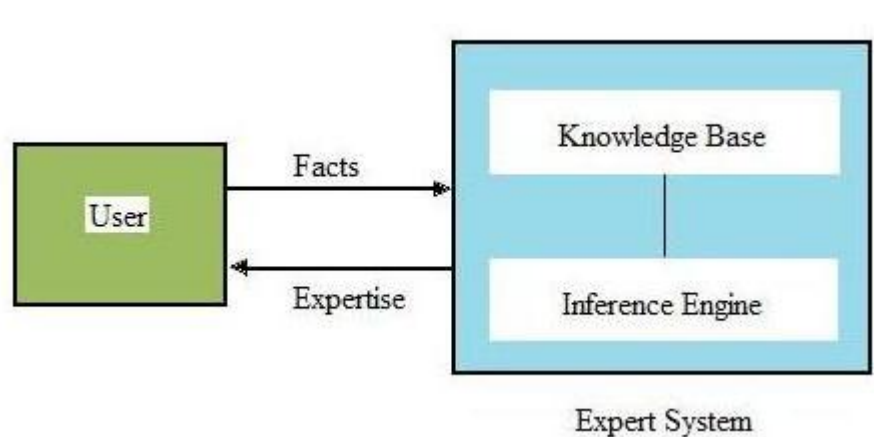


Figure 2.1: Expert System[4]

2.6 Genetic Algorithm

Genetic Algorithm is adaptive discovery research algorithm based on development opinions nature choice and genes. As like Gens display an intelligent development of a randomly investigate employed to find solution improvement trouble. Genetic Algorithm no signify randomly, in place of their use to advantage the past knowledge to straight the research inside of regional better carrying out between the research area.

The basis methods of Genetic Algorithms are fashioned to imitate methods in nature organized approach required for development, in special way these move behind the basic rules initial laid lower by Charles Darwin of “survival of the fittest” till in nature, competitions between persons for insufficient finances outcome in the fittest persons predominant above the not stronger ones [2].

2.6.1 Why Genetic Algorithms?

Genetic algorithms superior than usual Artificial Intelligence in that Genetic algorithm is greater strong. Different not newer Artificial Intelligence systems, They do not break easy ,if enters altered very little, or in the manner of person noise. And also, in researching a big condition-area, multi-modal condition-area, or n-dimensional outer side, a genetic algorithm may suggest with meaning advantages above greater normal research of making the best methods like linear program development, depth-first, breath-first implementation [5].

2.7 Fuzzy Logic

Fuzzy logic is approaching to compute based on “level of truth” prefer other the normal “true or false” (1 or 0) Boolean logic of the present time computer is based. Dr. Lotfi Zadeh’s opinion of the fuzzy logic was initially advancing the University of California at Berkeley in 1960s. Dr. Zadeh worked on troubles of computer understand of nature languages. Nature language (as greatest others activity in existence and in truth the world) is not easy to translate inside of the complete periods of 0 and 1. (if all things is basically expressible in binary points is a philosophic matter of having the value of following, yet in apply a lot of information people might desire to feed a computer is in a few condition in among and such, often, ends of data calculation). Fuzzy logic contains 0s and 1s, like farthest instances of truth (other “the condition of problems” other “true”) and also but contains some conditions of fact among because, for instance, the ending of comparing among two stuffs could not be “tall” other “short” but “38 of tallness”. Fuzzy logic appears nearer to the technique human brains operate. People summed information and shape a number of incomplete facts that people summed very far in much high from facts that successively, only if sure beginnings are went over, reason inevitable very far ends so like machine action as a result. A like type of methods is employed in artificial computer neural network and expert systems. It may aim to understand fuzzy logic like the technique use of reason in fact be employed and Boolean logic other binary is very easy exceptional instance of it [6].

2.7.1 Applying truth values

In figure 2.2 statements *cold*, *warm*, and *hot* are displayed functions drawing a temperature measurement [7]. A detail on system of measurement has tree “true values” for one each of three functions. The vertical row in figure displays specific temperature three arrows (true values) for measurement. Red arrow details to zero, temperature understand like “not hot”, orange arrow (detailing at 0.2) may defined it like In figure, the meanings of the expressions *cold*, *warm*, and *hot* are represented by functions mapping a temperature scale. A point on that scale has three “truth values” — one for each of the three functions. The vertical line in the image represents a particular temperature that the three arrows (truth values) gauge. Since the red arrow points to zero, this temperature may be interpreted as “not hot”. The

orange arrow (pointing at 0.2) may describe it as “a little warm” and the blue arrow (pointing at 0.8) “more or less cold” [10].

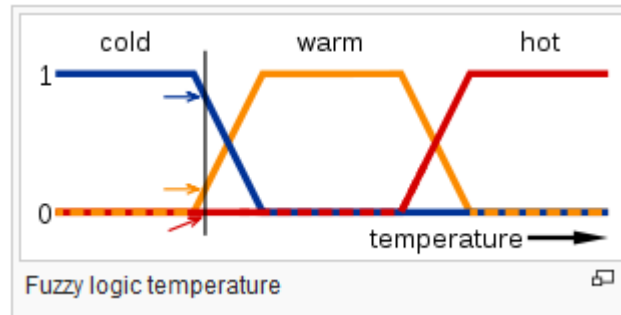


Figure 2.2 Fuzzy Logic[10]

2.8 Neural Networks.

It is a concept of processing data based on the way neurons in brains process information and communicates with each other. Neural computing is performed using Artificial Neural Networks (ANN). The brain consists of neurons. It is highly complex, on linear and parallel computer. In figure 2.3 below, Basic ANN architectures can be seen. It has the capability to organize the neurons so as to perform. Certain computations many times faster than the fastest digital computer in existence. Example applications of neural networks can be found in the fields of pattern recognition, perception and motor control while driving a car [11].

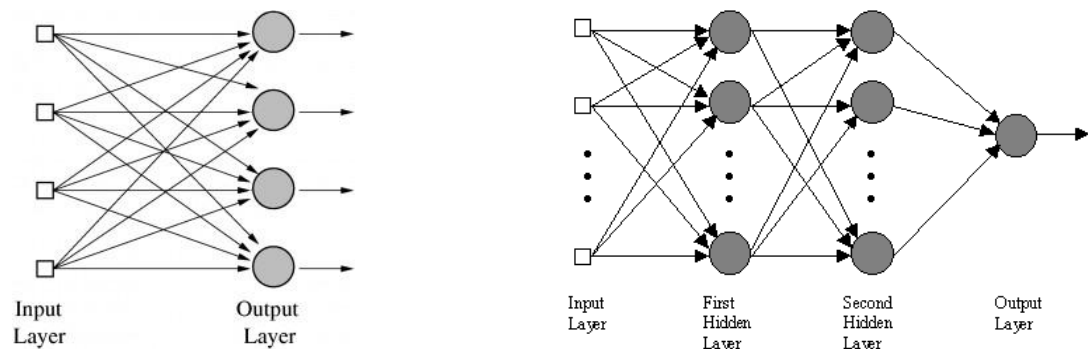


Figure 2.3: Basic Artificial Neural Network architectures (Single layer on the left and multi layer on the right)[11].

2.9 Summary

This chapter introduced briefly Artificial Intelligence and explained its branches like expert system, genetic algorithms, fuzzy logic and neural networks.

CHAPTER 3

ARTIFICIAL NEURAL NETWORKS

3.1 Overview

This chapter explains briefly Artificial Neural Networks. It describes the history and the back propagation Learning algorithm.

3.2 Brief History of Neural Networks

When we trace the history of Neural Networks, we can see that the origin of the field goes back to the 1940's when McCulloch and Pitts, for the first time introduced the neural network as a computing model. Next milestone comes when, in 1950's, Rosenblatt created the “perceptron”, the two-layer neural network that could do some learning by classification. The “perceptron” was good at pattern classification. But it was not a general classifier, for example it was having trouble in solving classic XOR problems. Although at that time, this limitation of the “perceptron” caused the decline of the neural network research, later became the foundation of the field. The decline in the research of neural networks lasted till early 1980s. In 1982 John Hopfield introduced “Hopfield networks”. Few years later the discovery of “back propagation” and the work of Kohonen on “Self-Organizing Maps” created new wave of the research in Neural Networks [8].

3.2.1 Neural Networks

Neural networks can be seen like machines that are designed to model the way in which the brain performs a particular task or function of interest. The neural network is usually implemented using electronic components or more commonly simulated in software. A neural network is massively parallel distributed processor that has the neural propensity of storing experimental knowledge and making it available for later use. Neural Network resembles the brain that knowledge is acquired by the network through a learning process. Inter neuron connections strengths known as synaptic weights are used to store the knowledge. The procedure used to perform the learning process is called Learning algorithm. The function of which is to modify the synaptic weights of the network in an orderly manner so as to attain a designed design objective [8].

3.2.2 Why use Neural Network?

Neural network can be used to take out models and notice fashions these are excessively complex detected by people or other computer techniques. A trained neural network can be idea of like an “expert” in the type of knowledge it has been given to analyze. This expert can then be used to furnish projections given new conditions of interest and response “what if” questions [8].

Other Advantages:

Adapt learning: A capability to learn how to make duties based the data given for training or first experiment.

Self-Organization: Artificial Neural Network represent of knowledge it takes in for the duration of learning.

Real Time Operation: Artificial Neural Network computations may be implemented in parallel and especial hardware devices are being planned and produced taking advantage of this capability.

Fault Tolerance in that manner Redundant Knowledge Coding: Partial destruction of a network leads to the corresponding degradation of performance.

3.3 Biological Human Brain

Human brain is setup to perform in the way we use it mostly in the first 2 years after birth, senses and the understanding of what is around us one all Learnt by the brain then. Afterwards more complex perceptions around us one slowly learnt other the years. This learning process is quite vital in the development of Neural Network and certainly differentiates Neural Network traditional computing systems.

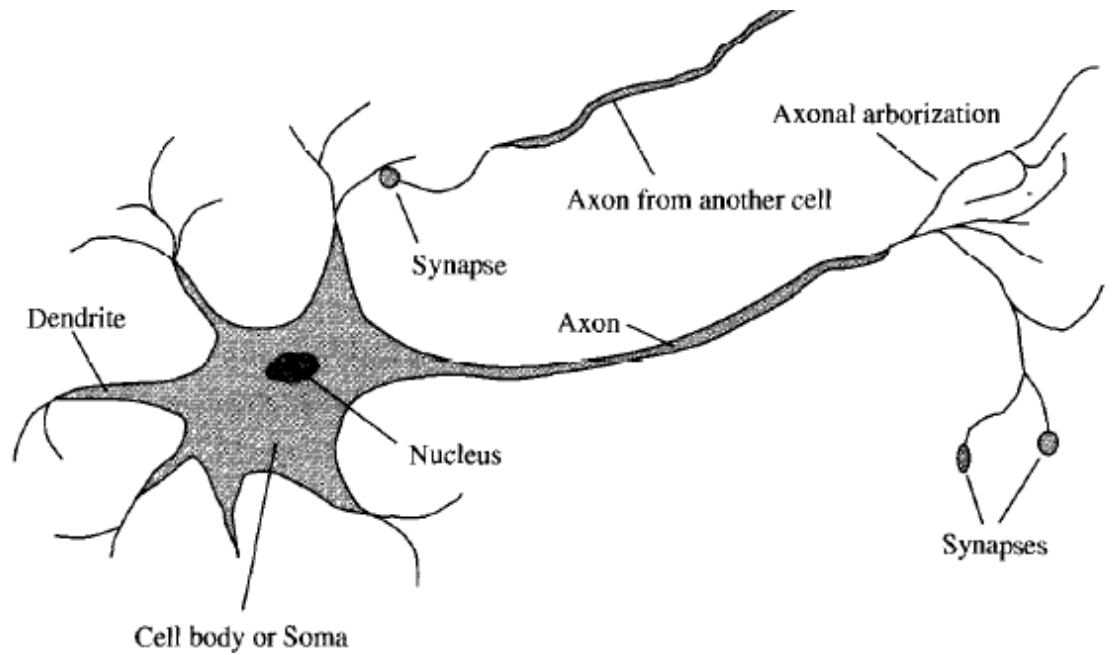


Figure 3.1: Biological neuron cell[11]

In the figure 3.1, a neuron cell is shown. The *synapse* is a small gap dividing neurons. Knowledge flows from one neuron to another across a synapse. Neurotransmitter leaves the axon terminal of one neuron and crosses the synapse to the dendrite of another neuron. The *axon* is the output lines of the neuron. Knowledge flows through the axon in electrical signal called action potential. The axon carries knowledge away from the cell body. Neurotransmitters are released from these terminals, sending a signal to the next cell. *Soma*, also known as the cell body, is the metabolic center of the neuron. The soma contains a nucleus. Dendrites carry information to the soma and axons carry information away from the soma. *Dendrites* are part of a neuron. They carry information to the cell body. The Artificial Neural Network is similarly made up of artificial neurons on a smaller scale [2].

3.3.1 From Human Neurons to Artificial Neurons

These neural networks by first aspiring to infer the important characteristics of neurons and their connections. Then they try to program a computer to reproduce these characteristics despite that because our information of neurons is unfinished and our computer force is restricted [9].

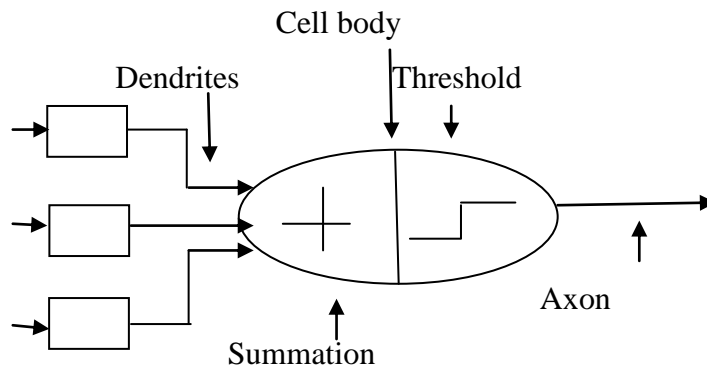


Figure 3.2: Architecture of an artificial neuron[9]

3.4 Artificial Neuron Models

Computational neurobiologists have built display computer method of neurons working to operate extensive computer of specific electrical devices in the brain. Like that Computer Scientists, human are extra stimulated in the usual properties of neural networks, free of how they are truly “applied” in the brain. This signifies that human can employ easier, not concrete “neurons” probably capturing essentially of neural computation and if they leave out collectively of how biological neurons work. In figure 3.2, a simplistic architecture of an artificial neuron can be seen. Human have applied pattern neurons in hardware like electronic circuits. Keep in mind thought that computers run much faster than brains. People can so that run rules big networks of easy method neurons like software simulated in intelligent time. This has advantages over having to se special “neural” computer hardware [10].

3.4.1 A Simple Artificial Neuron

Our basic computational element (model neuron) is often called a **node** or **unit**. It receives input from some other units, or perhaps from an external source. In the figure 3.3 the basic mathematical approximation of an artificial neuron can be seen. Each input has an associated **weight** w , which can be modified so as to model synaptic learning. The unit computes some function f of the weighted sum of its inputs:

$$y_i = f\left(\sum_j w_{ij} y_j\right)$$

Its output, in turn, can serve as input to other units.

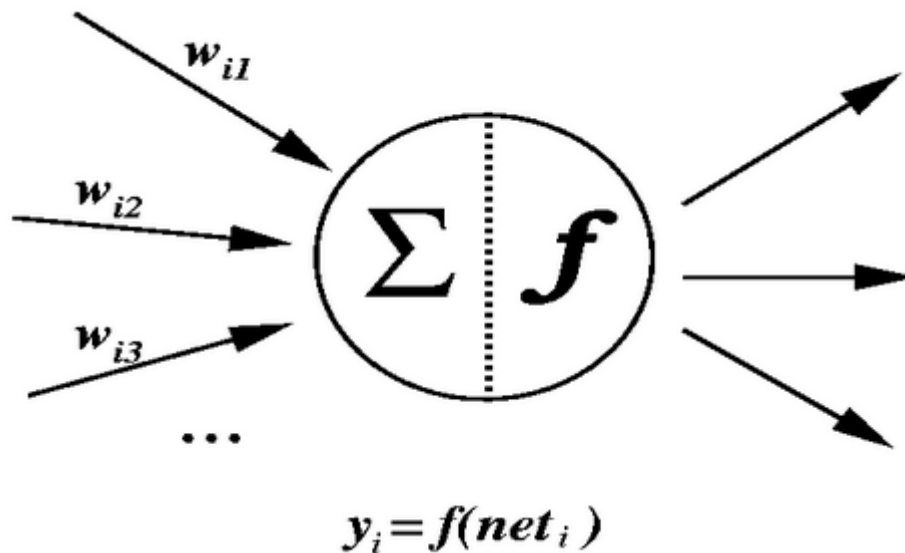


Figure 3.3:A simple neuron[10]

- The weighted sum $\sum_j w_{ij} y_j$ is called the **net input** to unit i , often written net_i .
- Note that w_{ij} refers to the weight from unit j to unit i (not the other way around).
- The function f is the unit's **activation function**. In the simplest case, f is the identity function, and the unit's output is just its net input. This is called a **linear unit** [10].

3.5 Learning in Artificial Neural Networks

Artificial neural network inspired by the biological nervous system, in particular, the human brain. One of the most interesting characteristics of the human brain is its capability to learn. We should note that our understanding of how exactly the brain does this is still very primitive, although we do still have a basic understanding of the process. It is believed that during the learning process the brain's neural structure is altered, increasing or decreasing the strength of its synaptic connections depending on their activity. This is why more relevant information easier to recall than information that hasn't been recalled for a long time. More relevant information will have stronger synaptic connections and less relevant information will gradually have its synaptic connections weaken, making it harder to recall. Although simplified, artificial neural networks can model this learning process by adjusting the weighted connections found between neurons in the network. This effectively emulates the strengthening and weakening of the synaptic connections found in our brains. This strengthening and weakening of the connections is what enables the network to learn. Learning algorithms are extremely useful when it comes to certain problems that either can't be practically written by a programmer or can be done more efficiently by a learning algorithm. Facial recognition would be an example of a problem extremely hard for a human to accurately convert into code. A problem that could be solved better by a learning algorithm, would be a loan granting application which could use past loan data to classify future loan applications. Although a human could write rules to do this, a learning algorithm can better pick up on subtleties in the data which may be hard to code for [11].

3.6 Learning Types

The learning process inside artificial neural networks is an outcome of changing the network's weights, with a few type of learning paradigms. The object is to learn a set of weight matrices that when implemented to the network should expectantly map any input output. to make corrections output.

3.6.1 Supervised Learning

In a supervised learning process, the input data and its corresponding output are presented to the neural network. The network will accord a defined low, change its

weights in order to be able to reproduce the correct output, when an input is given. Example of neural network based on supervised Learning.

3.6.1.1 The Perceptron

A perceptron model can be trained and can make decisions. During the training phase, pairs of input and output vectors are used to train the network. With each input vector, the output vector is compared with the desired output(target),and the error between the actual output and the target is used to update the weights.

3.6.1.2 Back propagation Algorithm

A multilayer network can be trained using the back propagation learning algorithm. This involves presenting pairs of input and output vectors. The actual output for a given input vector is compared with the target output. If there is no difference, the weights do not change. Otherwise, the weights are adjusted to reduce the difference. This learning algorithm uses a gradient search technique to minimize the cost function that is equal to the mean square difference between the target and the actual output .The network is initialized by setting random weights and thresholds.

3.6.1.3 The Hopfield Algorithm

A Hop Field network is essentially used with binary inputs. Weights are initialized using training samples. In the decision making phase the cost data is presented to the network at a certain time. Following the initialization the network iterates in discrete time steps using some mathematical function. The network is considered to have converged when the outputs no longer change on successive intentions.

3.6.2 Unsupervised Learning

An unsupervised learning process requires only input vectors to train the network. Once the input data is presented the neural network, the weights are adjusted in an order way according to some defined figure of merit. Below it is given examples of Unsupervised Learning processes.

3.6.2.1 Kohonen's Self-Organizing Maps

Kohonen suggested that one of the important Learning mechanisms in the human brain is the placement of neurons in an orderly manner. Kohonen's algorithm creates a feature map by adjusting weights from input nodes to output node in a two layer

network. The first layer is the input layer; the second is the competitive layers, and it organized as a two dimensional grid. The two layers are fully connected. Input vectors are sequentially presented to the input layer. Each unit computers the dot product of its weight with input vector value. This and its neighbor are allowed to learn.

3.6.3 Reinforcement Learning

Reinforcement learning is similarity to supervised learning in that a few feedback is given, Despite that in place of supplying a target output a prize is given based on how good the organize approach performed. The goal of reinforcement learning to make large the prize the system takes in through trial and error. This example tells powerfully with how learning works in naturally.

3.7 Adaptive Networks and Systems

In this section activation function Types will be explained.

3.7.1 Activation function Types

The most important unit in neural network structure is their net inputs by using a scalar-to-scalar function called “the activation function or threshold function or transfer function”, output a result value called the “unit's activation”. An activation function for limiting the amplitude of the output of a neuron. Enabling in a limited range of functions is usually called squashing functions . It squashes the permissible amplitude range of the output signal to some finite value. Some of the most commonly used activation functions are to solve non-linear problems. These functions are: Uni-polar sigmoid, Bi-polar sigmoid, Tanh, Conic Section, and Radial Bases Function (RBF). We did not care about some activation function such as identity function, step function or binary step functions as they are not used to solve linear problems [12].

3.7.1.1 Uni-Polar Sigmoid Activation function

The logarithmic sigmoid function is defined by its formula is given. In the figure 3.4 graph of the Uni-Polar Sigmoid is shown.

$$g(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid function is very useful to use in neural networks trained by back propagation algorithms. Because of function is very simple to recognize in an interesting to decrease the computation limit for training .The word sigmoid signifies “s-formed ‘logistic shape of the sigmoid graph [12].

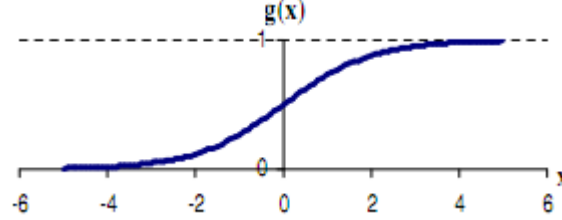


Figure 3.4: Uni-Polar Sigmoid Function[12]

3.7.1.2 Bipolar Sigmoid Function

Activation function of Bi-polar sigmoid function is defined by its formula is given. In the figure 3.5 graph of the Bipolar Sigmoid is shown.

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Bipolar sigmoid function is similarity to the sigmoid function .This activation function represented in Figure 3.3,it is advantages to use implementations that produce values in the range of [-1,1] [12].

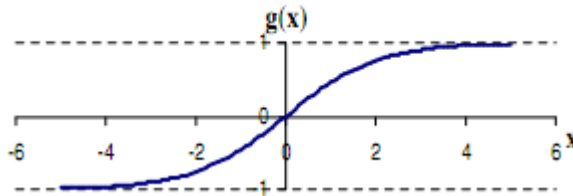


Figure 3.5: Bi-Polar Sigmoid Function[12]

3.7.1.3 Hyperbolic Tangent Function

Tangent function is simple described as the rate between the hyperbolic sine and the cosine functions enlarged as the rate of the half-difference and half-sum of two exponential functions in the points x and -x, is defined by its formula as follows:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

In the figure 3.6 graph of the Hyperbolic Tangent Function is shown. Hyperbolic Tangent Functions is similarity to sigmoid function. Its range outputs between -1 and 1 as represented in Figure 3.6. The following is a graphic of the hyperbolic tangent function for real values of its argument x [12].

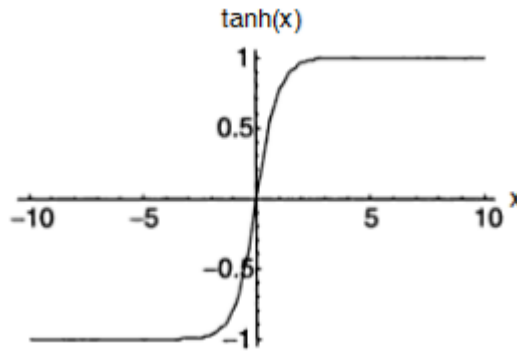


Figure 3.6: Hyperbolic Tangent Function[12]

3.8 Back propagation Artificial Neural Network

The back-propagation algorithm is applied in feed-forward artificial neural networks (ANNs). The nodes are organized in layers, and send their signals forward with errors propagated backwards. The network receives inputs by neurons in the input layer, and the output of the network is obtained by the neurons on an output layer. There may be several hidden layers. Each stage is connected to the next layer as shown in figure 3.7a below. The back-propagation algorithm uses supervised learning method, which means that we provide the algorithm with examples of the input and outputs we want the network to compute, and then the error is computed. The aim of the back-propagation algorithm is to reduce the error, until the ANN learns the training data. The training starts with random weights, and aim is to adjust them to arrive at minimal error. The number of layers and the number of artificial neurons per layer are important decisions to make when applying this architecture. The complexity between the input data and desired output determines the number of nodes in the hidden layer. Also, the amount of training data sets set an upper bound for the number of nodes in the hidden layer. This upper bound is calculated by dividing the number of input-output pair's examples in the training set by the total number of input and output nodes in the network. Then divide the result by scaling factor between five and ten [12].

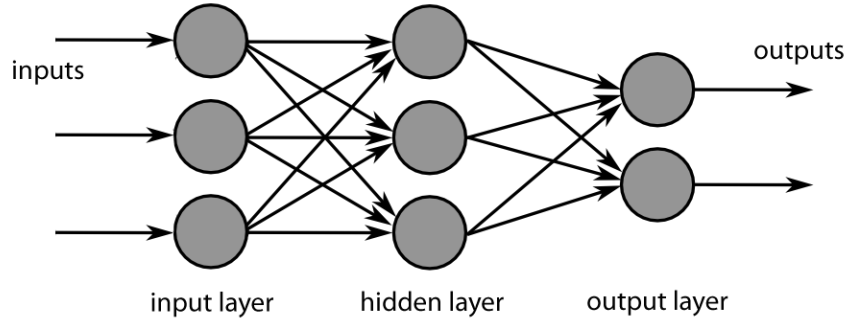


Figure 3.7a: A Feed Forward Back-Propagation Network [12]

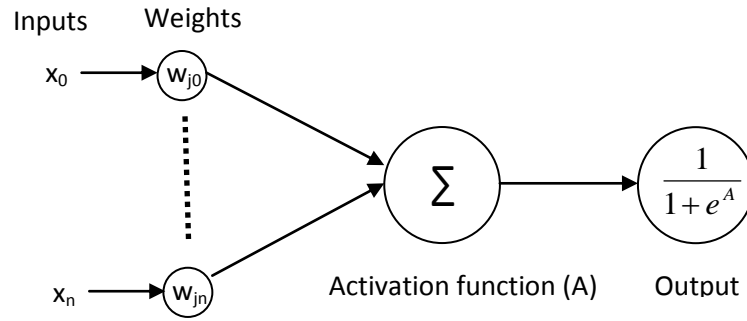


Figure 3.7b: Output calculation in the Feed Forward Back-Propagation Network

The activation function A_j is a weighted sum of the inputs x_i multiplied by their respective weights w_{ji} .

$$A_j(\bar{x}, \bar{w}) = \sum_{i=0}^n x_i w_{ji} \quad (3.1)$$

The activation depends only on the inputs and weights and the output function O is sigmoid function:

$$O_j(\bar{x}, \bar{w}) = \frac{1}{1 + e^{A_j(\bar{x}, \bar{w})}} \quad (3.2)$$

The output depends on the activation function, which depends on the weighted value of the inputs. This can be seen in the figure 3.7b above.

The aim of the training process is to obtain a desired output when certain inputs are given. Since the error is the difference between the actual and the desired output the error depends on the weights, and we need to adjust the weights in order to minimize the error (equation 3.4). The error function E_j of the output of each artificial neuron is given as:

$$E_j(\bar{x}, \bar{w}, d) = (O_j(\bar{x}, \bar{w}) - d_j)^2 \quad (3.3)$$

The square of the difference between the output and the desired output is used in order to have always positive result and higher precision. The error E of the network is the sum of the errors of all the artificial neurons in the output layer:

$$E(\bar{x}, \bar{w}, d) = \sum_j (O_j(\bar{x}, \bar{w}) - d_j)^2 \quad (3.4)$$

The weights are adjusted depending on their impact on the error E using the method of gradient descent:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \quad (3.5)$$

Equation (3.5) is applied until we find appropriate weights resulting to minimum error. The goal of the back-propagation algorithm is to find the derivative of E with respect to w_{ji} and since we need to achieve this backwards. First, we need to calculate how much the error depends on the output O_j , which is the derivative of E with respect to O_j from equation (3.3):

$$\frac{\partial E}{\partial O_j} = 2(O_j - d_j) \quad (3.6)$$

We then calculate how much the output depends on the activation, which also depends on the weights from equation (3.1) and equation (3.2):

$$\frac{\partial O_j}{\partial w_{ji}} = \frac{\partial O_j}{\partial A_j} \frac{\partial A_j}{\partial w_{ji}} = O_j(1 - O_j)x_i \quad (3.7)$$

From equations (3.6) and (3.7):

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial w_{ji}} = 2(O_j - d_j)O_j(1 - O_j)x_i \quad (3.8)$$

The necessary adjustment to each weight is gotten from equations (3.5) and (3.8):

$$\Delta w_{ji} = -2\eta(O_j - d_j)O_j(1 - O_j)x_i \quad (3.9)$$

Equation (3.9) is applied in training an ANN with two layers. Some additions will be made in order to train the network with one extra layer. If we want to adjust the weight of a previous layer, we first calculate how the error depends on the input from the previous layer. Here we use v_{ik} as the adjusted weight, and then we simply replace x_i with w_{ji} in equations (3.7), (3.8), and (3.9). We then determine how the error of the network depends on the adjustment of v_{ik} .

Thus we have:

$$\Delta v_{ik} = -\eta \frac{\partial E}{\partial v_{ik}} = -\eta \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial v_{ik}} \quad (3.10)$$

Where:

$$\frac{\partial E}{\partial x_i} = 2(O_j - d_j)O_j(1 - O_j)w_{ji} \quad (3.11)$$

Assuming that there are set of inputs x_i into the neuron with v_{ik} and then we have from equation (3.7):

$$\frac{\partial x_i}{\partial v_{ik}} = x_i(1 - x_i)v_{ik}$$

If we want to add yet another layer, we can do the same, calculating how the error depends on the inputs and weights of the first layer (equation 3.3) [20].

3.9 Summary

In this chapter a brief discussion on Artificial Neural networks is given and also back propagation algorithm and its formulas, some Learning algorithms (supervised and unsupervised) are explained in detail.

CHAPTER 4

ARTIFICIAL NEURAL NETWORKS APPLICATIONS FOR LEAVES RECOGNITION

4.1 Overview

This reports provided brief historical leaves recognition and the different experiments in testing performance of the foliage plant identification system, which incorporated shape, vein, color, and texture features and used as a classifier. The result shows that the combination of the features can improve the performance compared to the original work, testing.

4.2 Historical overview of Leaves recognition

There has been substantial work in recent years in the field for leaf biometric recognition. Initially it was approached by Petry. He classified weed species based on shape and structure of leaves to automatically. This morphological feature extraction technique has been used many times. (Stephen Gang Wu, 2007) used twelve morphological features (including vein features) and a neural network to achieve 90.3% classification accuracy. By using eleven morphological features Knight (2010) achieved a classification accuracy of 80%. This approach was implemented as a mobile application designed for field guides. S. Prasad (2011) has worked on leaf recognition using support vector machine with relative sub image based features. T. Beghin (2010) used a number of morphological features and a fuzzy surface selection technique to achieve 99% classification accuracy even with leaves that are deformed and oriented incorrectly. Madhusmita Swain (2012) performed plant classification by using the full color of the leaf in conjunction with support vector machine achieved 95% accuracy. However, this has the disadvantage that dry leaves cannot be used as they will be a different color from a leaf that has just been picked. Ehsanirad (2010) used a Gray-Level Co-occurrent Matrix (GLCM) and Principal Component Analysis (PCA) to achieve classification accuracies of 78.46% and 98.46% respectively. Using a Probabilistic Neural Network with, K. Singh (2010) achieved 91% classification accuracy. J. S. Cope (2010) compared four different texture methods for classification Gabor filters, Fourier descriptors, Co-occurrent matrices and Gabor Co-occurrences to achieve classification with accuracies of 50.78%, 82.42%, 69.14% and 85.16% respectively. Finally, some researchers have combined both morphological and texture-based techniques. T.

Beghin (2010) worked on Contour signature method for shape classification and the Sobel operator for texture classification. The result was a classification rate of 81.1%, significantly better than either of the methods when used alone. The problem with a number of these techniques is that they require some manual intervention such as correctly orienting the image or identifying the end points of the leaf's main vein. S. R. Deokar (2013) has worked on leaf recognition by extracting 28 and 60 Feature point. These features are extracted by vertical and horizontal splitting of the leaf images. ANN is used to compare performance of leaf recognition [13].

4.3 Why Leaves Recognition

Plants are an inherent of ecosystem. A lot of plant samples are into the danger of annihilation. Plants are very important beneficial for humanity being and other living vitals. Plants are beneficial to nourishment, to medical and also into a lot of industries. Plants Identifying supports insure to protective and survive of all natural life. Plant identification can be implemented a lot of various methods using the plant's leaves. . Classify plants to leaves are useful, Leaves are a lot of easily opportunity the other biometric elements the same as flowers are opportunity for duration of time. Different Biometric qualities of leaf as the same as color, venation, tissue, shape, in Plant classification needs to use. Identification guide is taking more time and costly. Leaves can classify as on color that inclusion semblance among both of two images the support of color, but color classification is related impression of sunlight on season [13].

4.4. Find the leaf venation pattern

Canny edge detection method is utilized for finding the venation pattern of the leaves. In the first stage of this method the leaf images are smoothed in order to remove the noise. Next stage involves finding the gradients with high magnitude by local maxima for identifying veins. Then veins are highlighted where high values of spatial derivatives are found. Edge candidates are selected after double thresholding. Finally veins are detected by eliminating veins that are not connected to strong veins [13].

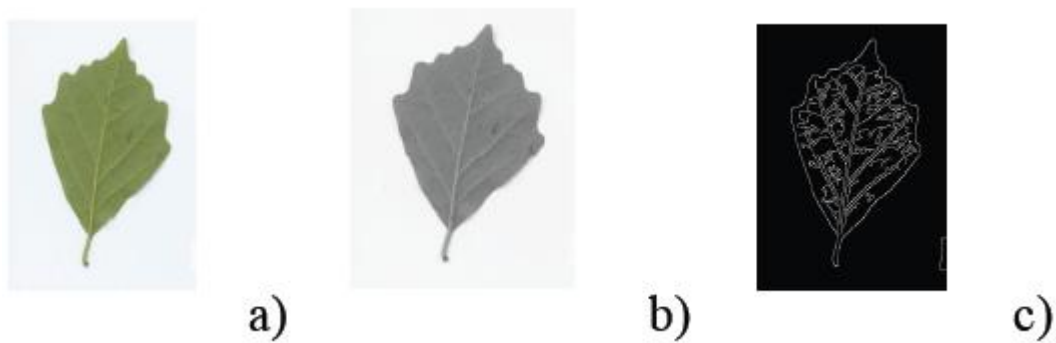


Figure-4.1: Leaf Images a) original b) gray scale c) venation pattern[13]

Leaf images are converted to gray scale images to avoid the potential negative effects of the variations in the color caused by the sunlight.

4.5. Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features

Through texture feature we can identify and classify the diseases in plants and can find the diseases in the initial stage. and with the help of pest control we can control the disease to some extent the reason why we misclassify the diseases are as follow. The symptoms of the affected plant leaves keep on changing (at the initial stage, tiny, dark brown to black spots, then letter on, it has the symptoms of withered leaf, black or part leaf deletion). If we want to remove the disease identification rate at various stages, the training samples can be increase in number and shape feature and color feature along with the optimal features can be given as input condition of disease identification. Examples of leaves with some diseases like early scorch, yellow spots, brown spots, late scorch, bacterial and fungal diseases are shown in Figure 4.

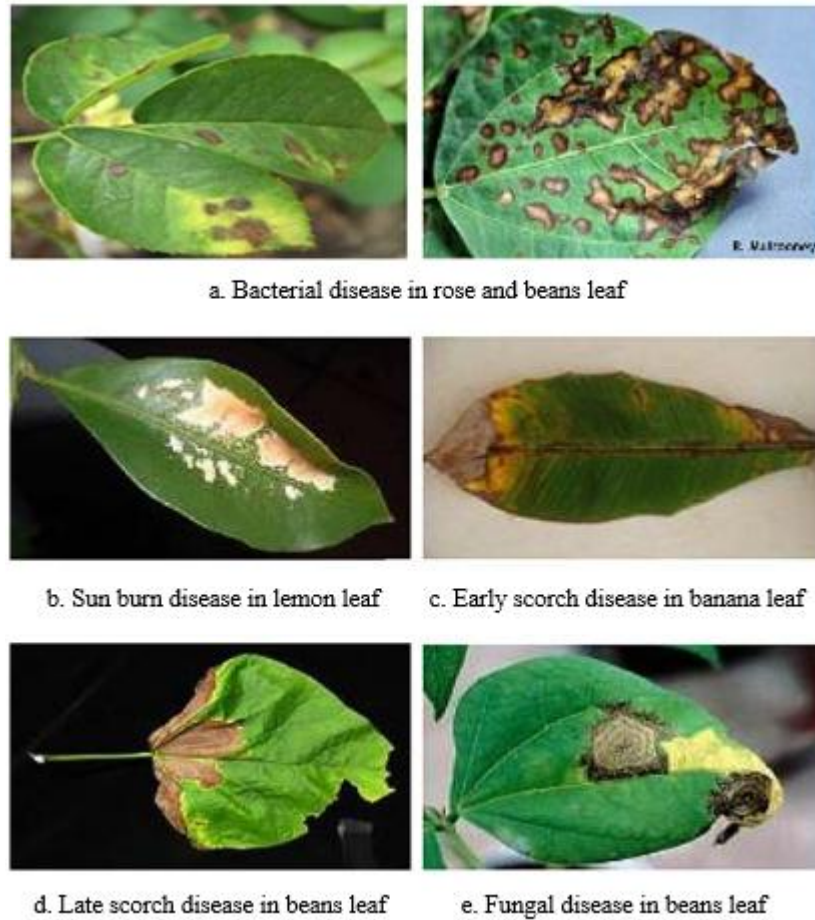


Figure 4.2: Sample images of infected leaves [14]

If we take some sample of leaves we can find out that they contain with various diseases like early scorch, yellow spots, brown spots, late scorch, bacterial and fungal diseases. Just taking as a sample, a rose leaf that is infected by bacterial disease is given as input to the algorithm. Color transformation structure on the input image can be performed and the results can be seen. Then the green pixels are masked and removed using a specific threshold value. Then the R, G, B components are mapped to the thresholded image. These steps are shown in Figure 4. Table 1 lists the set of leaves that are affected by various diseases [14].

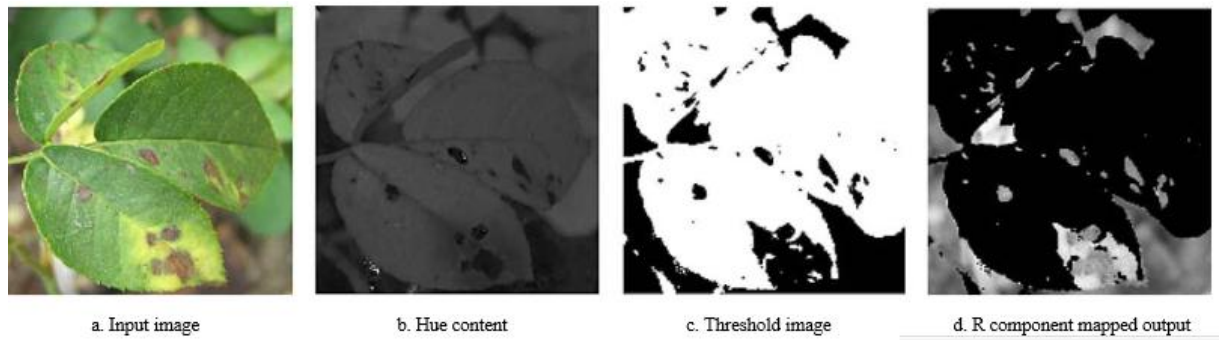

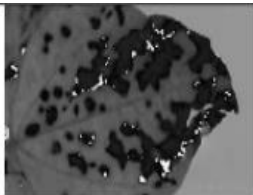

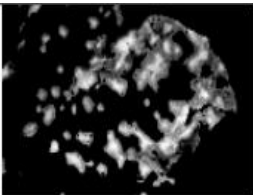

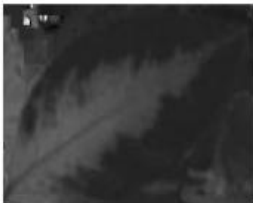

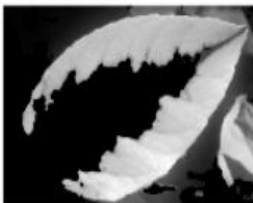





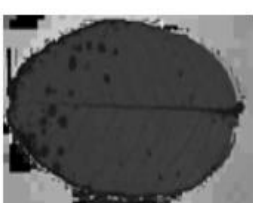




Figure 4.3: Detection of infected region for a rose leaf [14]

Table 4.1: Detected diseased region of various leaves [14]

Species mapped out	Input Image	Hue Content	Thresholded Image	R component
				
				
				
				

4.6 Disease Detection and Diagnosis on Plant using Image Processing

Plants get different diseases which decreases their productivity. These diseases can be identified according to their color, texture, and shape. Diseases of plants can occur in leaves stem. These diseases can cause due to virus, fungi, bacteria. They can be transmitted through insects. These diseases can be detected by Support Vector Machine (SVM), Artificial Neural Network (ANN), Back propagation (BP) Network, Probabilistic Neural Network (PNN), Radial Basis Function (RBF) Neural.



Figure 4.4: Infected leaf of cotton plant [15]

Farmers are qualified critics of the diseases, because of their experiences. But in fact they are not precise and correct sometimes. If farmers cannot control the disease then they call the experts, but this can be time consuming. Mostly the disease can be on leaves and on stem of plants. These diseases can be viral, fungal, bacterial diseases infected by insects, nematodes, rust on plants. It is very important duty for farmers to learn these diseases very early. Figure 4.4 shows the cotton plant disease which reduces the productivity. Precise and fast system to detect the diseases is needed [15].

4.7 Neural Network Application on Foliage Plant Identification

In some researches for leaf recognition, color is not used as a feature. The first reason is the fact that only green colored leaves are used. But for plants with leaves having interesting shapes and patterns color and texture properties should be included in recognition. As for instance, *Epipremnum pinnatum "Aureum"* and *Epipremnum pinnatum "Marble Queen"*, they almost have the same as texture, same shape, but their color is different from each other. Mixture of shape, texture, color characteristics should be included in the recognition for better results. For example in [12], three kinds of geometric and Fourier Transform characteristics were used to capture shape characteristic. Texture characteristics are removed from GLCMs, skewness was used to display color characteristics, and vein characteristics were added to make better identification. The identification system utilized Probabilistic Neural Network (PNN) as a classifier [12].

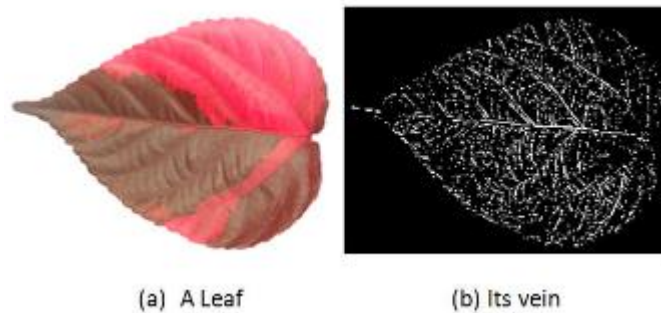


Fig 4.5: Illustration of vein processed by using morphological operation [12]

4.8 Early Detection of Pests on Leaves

Different image processing methods were used to spot and extract the pests in the captured image by early detection and extraction system. It's describing a camera filter that identifies pest species. The system then clarifies efficient methods to eradicate the threat. The procedures was used to take out of the detected things from the captured image is easy way with using shapeless similar segmentation. After that on processed image variant texture and color features are extracted. In the end, the characteristic values are fed as input to aid Vector Machine classifier, and then permit us to truly identify the insects and leaves .This is first step an important to identification of insects to find the corresponding to solution problem, in the next time to detect the different kinds of insects a single advanced method [16].



Figure 4.6: (a) Input image [16]

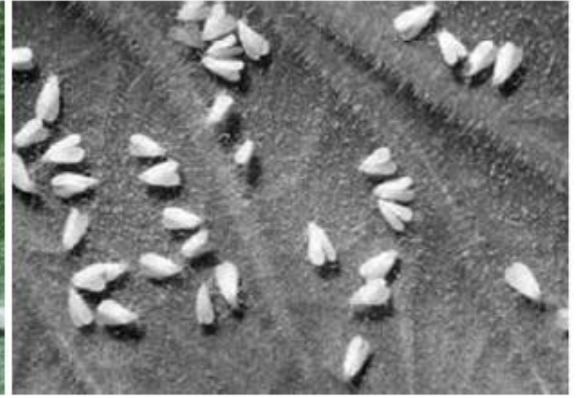


Figure 4.7: (b) Gray scale [16]

4.9 Plant Leaves Disease detection using Image Processing Techniques

Agricultural products are affected by disease plants. This phenomenon is one of the major reasons of economic production losses. Furthermore, diseases reduce the quality and quantity of agricultural products. This study presents the simple detection system for plant leaves diseases, by using image acquisition, image pre-processing, features extraction and neural network based classification. Further discussions on similar systems can be found in [14].

4.9.1 Plant diseases analysis and its symptoms

- **Bacterial disease symptoms**

The diseases is depicted very small pale green spots early come into view as water soaked. The lesions become bigger after that seem like dry dead spots in figure 4.8 e.g. Bacterial leaf spot have black water-soaked and brown spots on the leaves. It can be yellow halo, usually exactly the same size. Under dry positions the spots can appear as speckled.



Figure 4.8: Bacterial leaf spot [14]

- **Viral disease symptoms**

Viruses are not easy to diagnose between all plant leaf diseases. In agricultural product viruses no revealing signs can be reacting quickly see and frequent readily not understand nourishing inadequacy and herbicide act of injuring. Leafhoppers, aphids, whiteflies, cucumber beetles and whiteflies pests are often carries disease example Mosaic Virus, as appear yellow, green stripes other spots on leaves. in the following figure 4.9. Leaves might be curled, growth, wrinkled and stunted.



Figure 4.9: mosaic viruses [14]

- **Fungal disease symptoms**

In all plant leaf disease, Fungal reasoned with fungus some of them in show figure 4.10 below Late blight reasoned by the fungus *Phytophthora infesters* display in figure 4.10 (a). First seems over very small, older leaves as water-soaked, gray-green spots. At what time fungal disease, grow up make darker in color. This spots become dark and after that fungal increase forms on the underneath. Early blight is reasoned by the fungus *Alternaria solani* display in figure 4.10 (b). First, seems over very small, older leaves like small brown spots with concentric rings that form a bull's eye model. At what time disease grows up, it propagates outside on the leaf surface reasoning leaf to turn yellow. In downy mildew yellow to white patches on the upper surfaces of older leaves occurs. These areas are covered with white to grayish on the undersides as shown in figure 4.10 (c).



(a) late blight (b) early blight (c) downy mildew

Figure 4.10: Fungal disease on leaves [14]

4.10 Summary

In this chapter, by analyzing the characteristics of plant leaves, it is presented several feature extraction methods and classification algorithms. Among those methods are the optimized BP neural network and Probabilistic Neural Network, Support Vector Machine (SVM), Artificial Neural Network, Radial Basis Function (RBF) Neural Network.

CHAPTER 5

LEAVES RECOGNITION AND EXPERIMENTAL RESULTS

5.1 Overview

A leaf recognition process must discuss two basic points; the fundamental of the most important special features of the leaf, and the recognition of these leaves or the classification of them. In neural networks, the network tries to classify the sets of leaves based on their color concentration without doing any mathematical or statistical studies.

This chapter introduces a detailed discussion about the back-propagation based leaf recognition process.

The use of the back propagation in the recognition of different leaf images is discussed and introduced. 27 different leaves are used in the identification of 27 trees. These are Alligator-pear, apple, benjamin1, benjamin2, bougainvillea, Cherry laurel, duranta-tree, False-poplar, gale, carob, Israel rubber, kamkuat, mandarin orange, new word fruit, okuloptus, oleander, orange, passion-fruit, pitosporum, solanum-rantonetti, olive, bottle-brush, ficus, Laurus nobilis, lemon, Psidium, the spindle tree, the sample of this thesis.

For the training of neural networks, 9 sets of photos for all the 27 different tree leaves were fed to the network. These 9 sets include the original photos in addition to 8 different noises or noise level. Mainly, the used noises were Gaussian noise, salt and pepper, speckle, and Poisson noises with different noise levels. After the training of the network all the training sets are passed through the network to check the efficiency of the training process. For the test of the neural networks three different testing processes are arranged. Each process contains the four types of the mentioned noises with a preset noise concentration. The set Level III contains the higher noise ration compared to the other two sets. The sets of real and noisy images are preprocessed and then provided to the ANN using the back propagation learning process in the training stage. After that a simple test is applied to check the ability of network to recognize processed leaf photos. This chapter includes the methodology of the research and the different steps of it; starting by collecting the data base and ending with the testing stage. The implementation of the noise and processing of the images is an important stage in the recognition procedure. The processing of the

images is done by using MATLAB which includes adding the noise to the photos in addition to resizing and changing the type of images to reduce processing expenses. The pre-processing stage is a very important stage for a prosperous recognition rule. The choice of image size is important as it affects directly the results of the program in addition to the running time.

5.2 Structure of the System

Below the block diagram of the process is given. It can be seen the preprocessing stages for the image before it is fed to the NN. Figure 5.1 describe this structure of design system. The original image that is acquired first converted to gray scale image and then scaled to 50x50 size. After that noise is added. Finally the resulting 50x50 pixel matrix is fed to input layer as column based. So the network has input layer with 2500 nodes. After some experiment it has been found the optimum range for the number of neurons for the hidden layers as 200 and 240. One of the 27 output neurons when it fires, after the processing, specifies the matched leaf number.

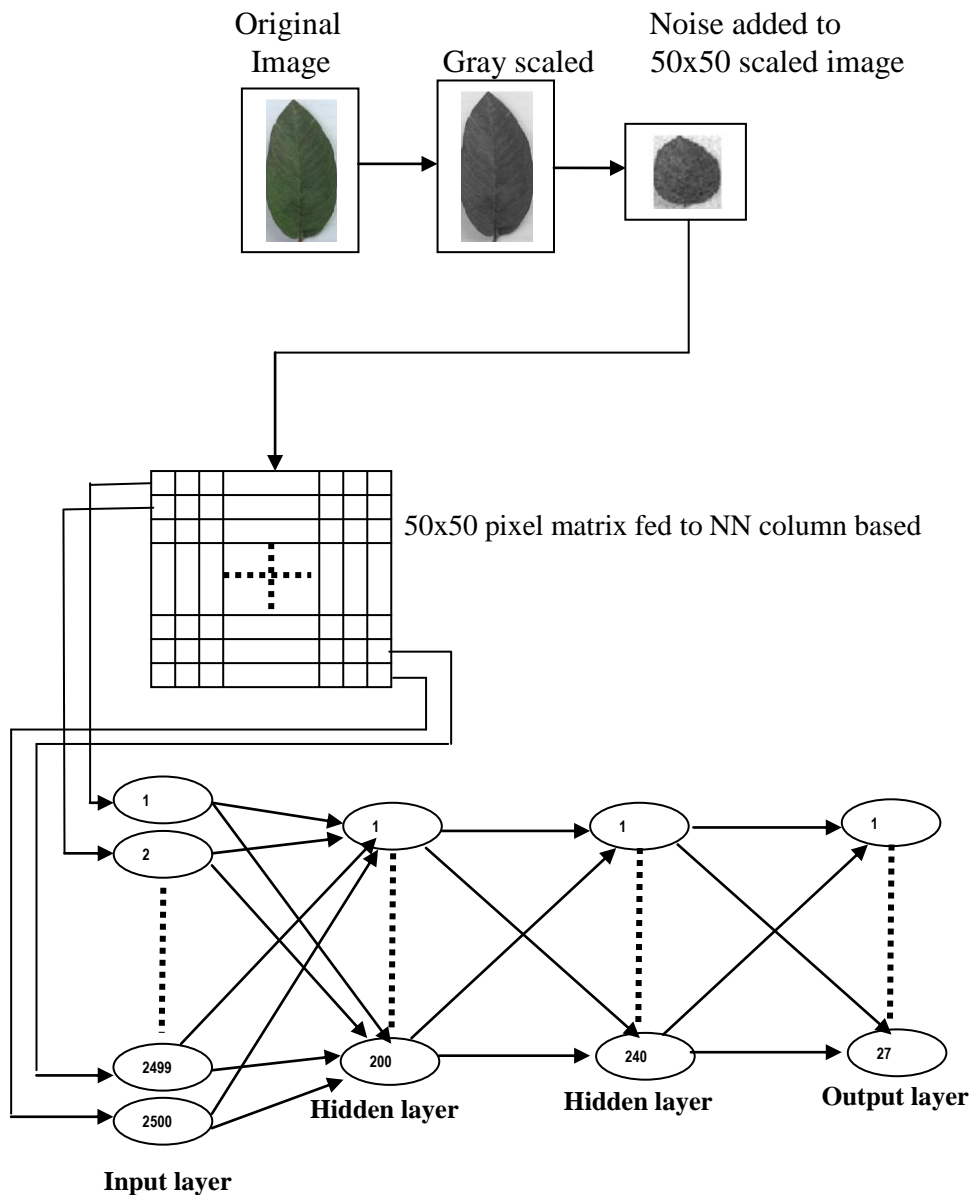


Figure 5.1: Block diagram of the System

5.3 Methodology

Two types of experiments were designed. The first one was on images with noise. For the first experiment single leaf image of each plant was used to add different types and different amounts of noise. 27 different leaves from native plants of the Cyprus were collected and their images were digitized. Images of the leaves and their names that were used in the first experiment can be seen in Appendix A.

The second one was on the subset of Flavia Database¹. For the second part 10 different copies of each leaf was used without adding any noise. 20 different plants were chosen from Flavia Database. Images of the leaves that are selected from Flavia Database for the second experiment can be seen in Appendix A. The common names of the plants are also listed in Appendix A. The system was tested to see how well the recognition can be with variations in the leaf shape, size and orientation.

After data collection stage was finished, the pre-processing stage starts with applying different steps of processing to prepare the different data base sets to be used as training and testing inputs and targets of the neural networks. Figure 5.1 represent a block diagram of the pre-processing steps before sending the data to the neural networks.

¹ <http://flavia.sourceforge.net/>

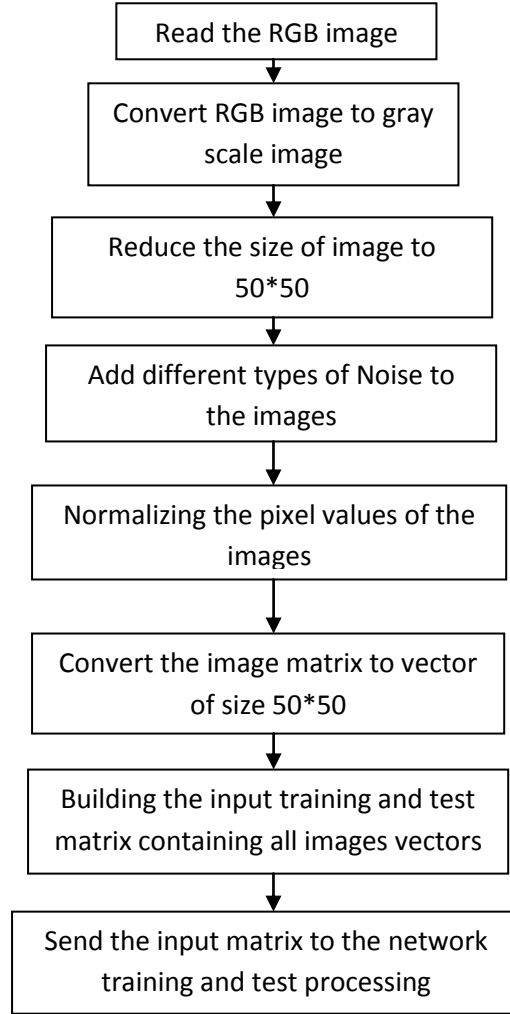


Figure 5.2: Block diagram of the preprocessing phase of the training and test images.

The data base represented RGB images of the 27 different tree leaves which are the subject of our study. These 27 photos were initially read using MATLAB in RGB format. The RGB images were converted then to grey scale image in order to reduce the calculation cost and simplify the training and test process. These gray scale images are then resized to fit the function of neural network. As large size images will take very long time to be trained using neural networks. The size of 50*50 was used because it is suitable in terms of calculation cost and keeps enough description of the features of images. In the next stage the images were copied into two groups, one for training and one for test. The training set contains 9 copies from each of the leaf images, giving us $9 \times 27 = 243$ samples. Noise was added to these images to train the network for different possible images that can be fed to it.

Nine copies from each were taken and different types of noise were added to them.

- The first copy has Gaussian noise with mean 0.02 and variance 0.005,
- The second copy has Gaussian noise with mean 0.02 and variance 0.01,
- The third copy has Poisson noise,
- The fourth copy has Speckle noise with variance 0.04,
- The fifth copy has Speckle noise with variance 0.08,
- The sixth copy has Salt&pepper noise with density 0.01,
- The seventh copy has Salt&pepper noise with density 0.04,
- The eighth copy has Salt&pepper noise with density 0.08,
- The ninth copy is the original image without noise.

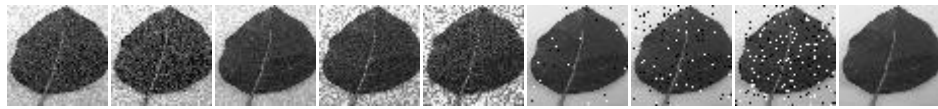


Figure 5.3: 50x50 Training set images for the Alligator pear. The order is from left to right as it is explained above. The first image on the left has Gaussian noise with mean 0.02 and variance 0.005. The last one, the rightmost image is the original image.

The test set contains three sets containing 4 noisy copies of each leaf image, giving us $4 \times 27 = 108$ samples to test for each noise group. The first set, Level I, is low-level noise set with the following 4 types and levels of noise:

- Gaussian noise with mean 0.03 and variance 0.01,
- Salt&pepper noise with density 0.1,
- Speckle noise with variance 0.12,
- Poisson noise,

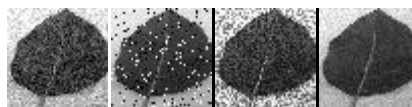


Figure 5.4: 50x50 Level I Test set images for the Alligator pear. The order is from left to right as it is explained above. The last one, the rightmost image is with Poisson noise.

The second set, Level II, is mid-level noise set with the following 4 types and levels of noise:

- Gaussian noise with mean 0.03 and variance 0.02,
- Salt&pepper noise with density 0.12,
- Speckle noise with variance 0.22,
- Poisson noise,

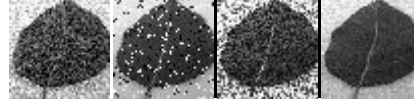


Figure 5.5: 50x50 Level II Test set images for the Alligator pear. The order is from left to right as it is explained above. The last one, the rightmost image is with Poisson noise.

The third set, Level III, is high-level noise set with the following 4 types and levels of noise:

- Gaussian noise with mean 0.03 and variance 0.03,
- Salt&pepper noise with density 0.15,
- Speckle noise with variance 0.28,
- Poisson noise,

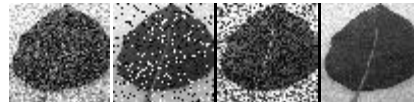


Figure 5.6: 50x50 Level III Test set images for the Alligator pear. The order is from left to right as it is explained above. The last one, the rightmost image is with Poisson noise.

Table 5.1: Below are shown noise parameters for testing images. Same Poisson noise added to all levels.

	Level I	Level II	Level III
Gaussian	m=0.03 var=0.01	m=0.03 var=0.02	m=0.03 var=0.03
Salt&Peppper	d=0.1	d=0.12	d=0.15
Speckle	var=0.12	var=0.22	var=0.28

Level I has images with low level noise, but higher than the training images, while Level II has more noise ratio, and the Level III group contained the highest level of noise. As the neural networks perform better when their inputs are limited to a defined period of values, the pixels of the different sets of images were normalized to such that their values be limited to the interval $[-1,1]$. Each one of the images was converted to a vector containing 2500 elements to simplify the process of training and testing. Using the one dimensional representation of the images, the input matrix was built containing all training inputs and a target matrix was also prepared presenting one code for each image. The input and output matrices can then be sent to the network to complete the training process. Related matlab codes for the generation of these images and for the testing-training of the ANN, can be found in Appendix B.

For the second part of the thesis subset of 200 leaves (20 plants and 10 different leaves from each plant) from the Flavia Database is used to perform experiments. All the images are scaled to 50x50 gray scale images as in the first part. But in the second part of the experiments noise was not added. The same algorithm is tested to see the classification capacity. 7 images out of 10 are used for training and remaining 3 are used for testing. The version of matlab code that is used for the second experiment can also be found in Appendix B.

5.4 Database Collection

The first of all in this work was the data base collection. All the images were to collect from the trees of plant center in Nicosia. The photos of leaves of 27 trees were collected using a digital camera. A distance of arbitrary cm was chosen between the camera's lens and the leaves with perpendicular angle of shot. The photos were then matched up and processed using Photoshop program and Paint. The photos were arranged such that just the special features of the leaves appear in the picture. Any empty spaces or other special features were manually removed from the photos. The preprocessing stage include changing the images to gray scale images, resizing images, normalization of images data, and adding noise to images as explained earlier in this chapter. In the training process of the neural network 9 sets of normal and noisy images were used. The first set represented the normal images of the leaf. The other sets were noised using MATLAB program. Different types of noises were used in the training like Gaussian and salt and pepper noise.

After processing all images and preparing the input and output matrices, all special features of training examples are introduced to ANN. The next weights of neural network that are provided from training stage will be used to test the network. The results are compared to original classes to get study of the system.

5.5 General Experiment

Various experiments were applied on the sets of images until arriving suitable parameters of the neural network in the training process. In the training process of the neural network, 9 sequences of normal and noisy images were used. The first sequence displayed the normal images of the leaves. The other sequences were noised using MATLAB program. Different examples of noises were used in the training like Gaussian and salt and pepper noise. The MATLAB code used for adding different examples of noise is added in Appendix A, An Example of the original and noisy images (salt and pepper noise with density 0.08) is shown in figure 5.7. As we can see from the figure, the noise added to the training images was low.

The images of the figure 5.7 display the training example of the first, second, third, and fourth Leaves. These images were preprocessed before being fed to the network. The preprocessing of the images passes by different stages. The first stage after reading the image in RGB scale is to change it into gray scale image. The gray scale

image displays each pixel of the image by an unsigned eight bit integer (0-255). This count is the concentration of white color in the pixel. The pixel is black if its value is zero, increasing the value increase the white concentration until it maximum in the white color with the value of 255. In the RGB scale images each pixel is represented using 4 different values; each one of these three values represents the concentration of the

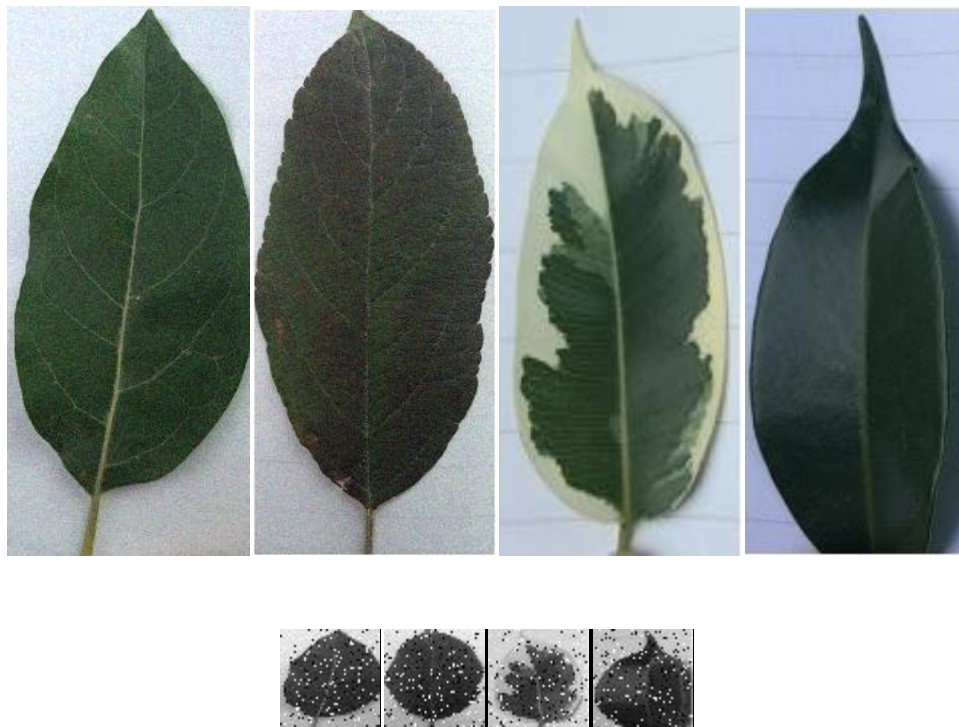


Figure 5.7: Sample of the images used in the training (Alligator_pear, Apple, Benjamin1, Benjamin2).

three base colors, red, green, and blue. Using the gray scale image reduces the image size by two thirds of its original RGB scale size. That operation reduces the calculation efforts continued by the program with absolutely no effect on the accuracy of the program.

After the process of changing image scale, the size of the image has to be reduced to an acceptable size to make easier the operations on smaller images in addition to reduce the processing cost of the program. Figure 5.8 shows the original RGB image

and gray scale image. In figure 5.9 it can be seen the resized (50x50) image of the same leaf.



RGB image



Gray scale image

Figure 5.8: Original RGB image and Gray scale image.



Figure 5.9: Resized gray scale image.

The resized image is after that normalized in order to limit the inputs of the ANN to $[-1..1]$ range. This operation reduces the processing costs and learning time. The processed images must be tested and fed to the neural network such that the one leaf

information is processed in each repetition. In the next repetition, an image of a different leaf should be represented to the network until the last image finish. After finishing all images, the operation must be repeated from the beginning until the network learns. In order to make easier the discussed operation, a small routine was written using MATLAB to provide each image's pixels in a vector. The vectors are then providing in a big matrix which will be fed column by column to the network. The desired output of the network was chosen according to the input vector, in other words to the order which the training images are put to the input vector. During the training of the neural networks a target error was chosen to be 1×10^{-4} , and the maximum epoch was chosen to be 1500. A stop condition of training is to reach one of these two parameters.

In the target matrix the columns presents

1. 'alligator_pear.jpg'
2. 'apple.jpg'
3. 'benjamin1.jpg'
4. 'benjamin2.jpg'
5. 'bottle_brush.jpg'
6. 'bougainvillea.jpg'
7. 'cherry_laurel.jpg'
8. 'duranta_tree.jpg'
9. 'false_poplar.jpg'
10. 'figus.jpg'
11. 'gale.jpg'
12. 'harrup.jpg'
13. 'israel_rubber.jpg'
14. 'kamkuat.jpg'
15. 'Laurus_nobilis.jpg'
16. 'lemon.jpg'
17. 'mandarinorange.jpg'
18. 'new_world_fruit.jpg'
19. 'okuloptus.jpg'
20. 'oleander.jpg'
21. 'orange.jpg'
22. 'passion_fruit.jpg'
23. 'pitosporum.jpg'
24. 'Psidium.jpg'
25. 'Solanum_rantonetti.jpg'
26. 'the_spindle_tree.jpg'
27. 'zeytin.jpg'

respectively.

5.6 Training process

After many trials using different training parameters, the best parameters of the neural networks were found and the back-propagation (for the training Gradient Descent with momentum and adaptive learning method was used) process was started with following parameters

Table 5.2: Training parameters of the ANN used for the first experiment.

Number of input neurons	2500
Number of neurons for the first hidden layer	200
Number of neurons for the second hidden layer	240
Number of output neurons	27
Learning rate	0.05
Momentum factor	0.9
Error tolerance	1e-4
Minimum performance gradient	1e-5
Training time	169 second
Max Epochs	1500 (reached)
Target error	5×10^{-7}

In the figure 5.6 the ANN architecture that was used can be seen. For the hidden layers logsig transfer functions are used whereas for the output layer linear transfer function is used for better resolution of the output values.

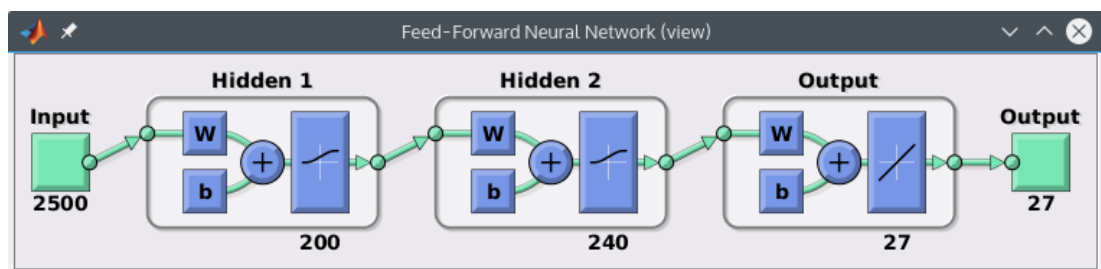


Figure 5.10: Curve of the MSE during the training.

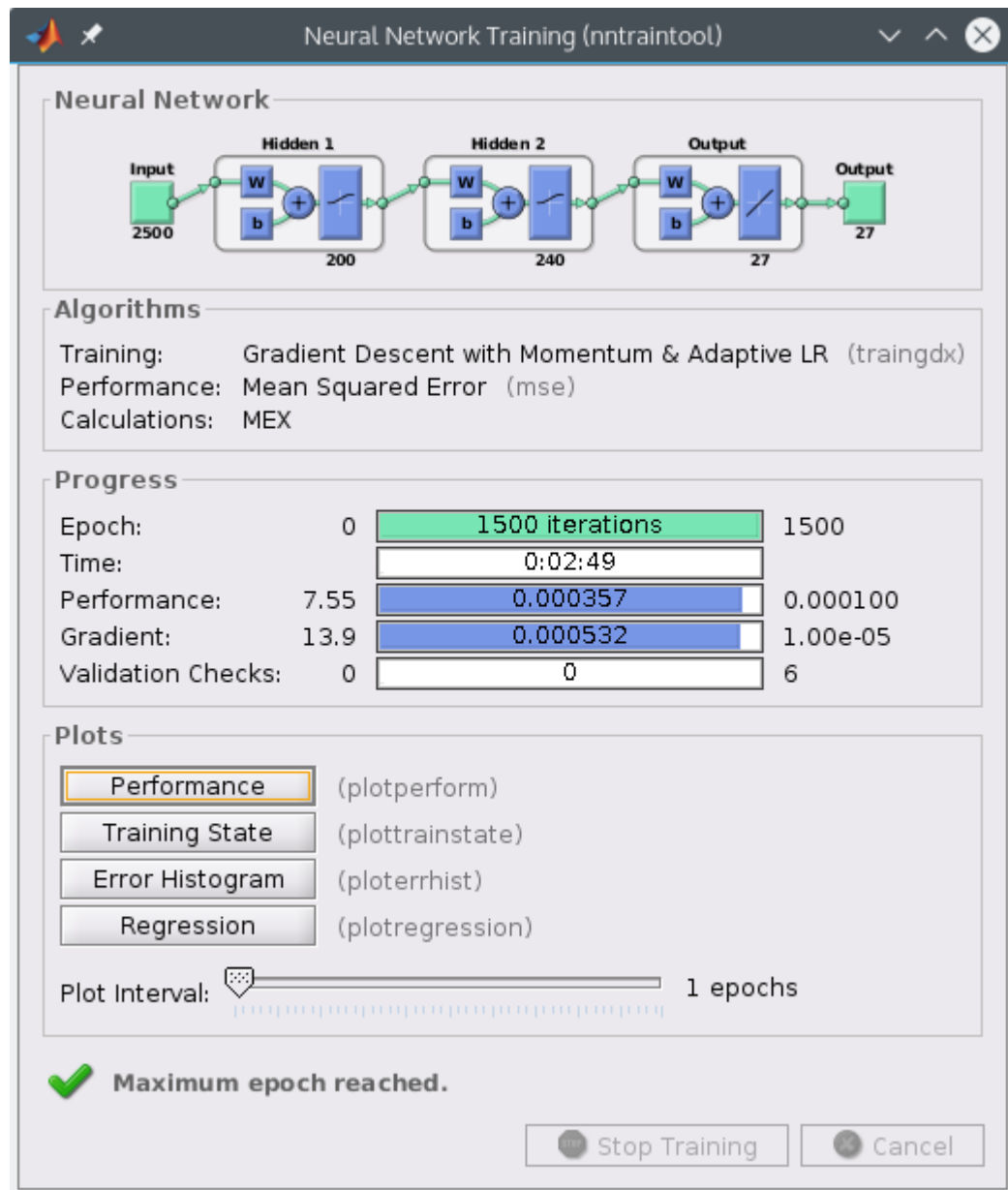


Figure 5.11: MATLAB interface of neural networks during training process.

A recognition ratio of 100% was obtained during the training and no images were misrecognized. The fact that the error goal was very small ameliorates the results during the training process.

After the end of training, single repetition test was applied to check the ability of trained network to recognize other than the training sequence. The test was applied with three different groups of original images with different noise parameters. Each sequence contains 4 leaf images of each of the 27 trees. Table 5.3 displays the testing results of the Level I, Level II and Level III comparatively.

Recognition rate for the test of the first set of images, namely for Level I, was 97.2% where 105 images out of 108 were recognized. Concerning the second set, namely Level II, 99 images out of 108 were recognized rightly with a rate of 91.7% which seems to be a good rate given the high noise conditions. In the test of the third set, which was Level III, the system was able to recognize 86 images out of 108 images contained in the set, achieving a rather good recognition ratio of 79.6% given the existence of the high noise ratio. Each of the Level I, II and III contains 4 different images each having different types of noise.

Table 5.3: Overall results for the Level I, II and III.

Level I	Level II	Level III
97.2%	91.7%	79.6%

For the Flavia Database Subset experiments 7 images are separated for training and 3 are separated for testing. Total 60 images are used for testing as there were 20 different plants and from each 3 images are separated. Some parameters are updated from the original code that was used for noise experiments for the Flavia experiments. Error tolerance (`net.trainParam.goal`) was set to $1e-7$ where as it was $1e-4$ for the noise experiments. Minimum performance gradient (`net.trainParam.min_grad`) was set to $1e-6$ where as it was $1e-5$. Finally epoch number (`net.trainParam.epochs`) was set to 2000 for the Flavia experiments. The recognition rate was 95% (57 out of 60 test images were recognized).

5.7 Some results from other researches

Below, in the table 5.4, it is given a collection of the recognition results along with the explanations about the method and the scale of the leaf database used in the research. The results shown in these researches are neither the best nor the worst results. But they are chosen on the bases of the diversity of the methods utilized in the recognition tests. The selection of these works is based on the work done by Amlekar et al [13].

Table 5.4: Some recognition results from different recent researches

Work	Method	Database	Rate
S. Gang Wu et al, 2007 [17]	Morphological features and PNN	32 different plant leaves	90.3%
S. Prasad et al, 2011 [18]	Support vector machine	23 different plant leaves	95%
A. Ehsanirad, S. Kumar, 2010 [19]	Gray-Level Co-occurrent Matrix (GLCM)	13 different plant leaves	78.46%
J. S. Cope et al, 2010 [20]	Gabor Co-occurrences	32 different plant leaves	85.16%
The implemented method (noise experiments)	50x50 leaf image fed to NN with Back propagation	27 different plant leaves	97.2%
The implemented method (Flavia Subset experiments)	50x50 leaf image fed to NN with Back propagation	20 different plant leaves	95%

5.8 Summary

In this chapter explanation about Leaves recognition is presented. The methodology for the training and for the testing of leaf images is explained. The recognition rates and results were also given and explained. This system used real-life images. The testing results show us this system can be used in real-time applications where detects leaves recognizes. This thesis illustrates that the characteristics of the selected parameters for neural network are feasible and practical in the classification of plant leaves.

CONCLUSIONS

Leaves recognition has been discussed in different scientific papers and researches. It can contribute strongly in the science of plants classification. This work has been carried out in the goal of introduction of leaves identification or classification using ANNs. The neural networks have proved their ability to give high efficiency in different applications. A leaf recognition process must discuss two basic points; the fundamental of the most important special features of the leaf, and the recognition of these leaves or the classification of them. In neural networks, the networks tries to classify the sets of leaves based on their color concentration without doing any mathematical or statistical studies. From the experiments carried out in this thesis and the results obtained we conclude that the use of the neural network for leave recognition and plants classification was successful. The application of different noise on the leaves' images has led to different recognition rates. Different experiments including training and test of networks have been carried out in this work. In the training process 9 set of images were prepared and fed to the neural network. The process of back propagation has been started until an acceptable error was achieved. For the purpose of testing the obtained network's efficiency with different images out of the training sets; 3 different groups of images were prepared with different noises. These groups were divided into Level I, II, and III containing 4 sets of images each. These three groups were then fed to the network and their results were obtained. 100% out of the 243 training images were recognized correctly; whereas 104/108 images were recognized from the first group of images. That shows that the recognition rate was 97.2%. In the Level II group, 99 images out of 108 were recognized rightly with a rate of 91.7% which seems to be perfect under the high noise conditions. In the test of the Level III, the system was unable to recognize 22 images out of 108 images contained in the set. The recognition ratio was 79.6% in this experiment, which is considered very high under high noise ratio parameters of images. For the Flavia Subset experiment recognition rate was 95%. 57 of the 60 testing images were recognized correctly. Flavia database subset has provided a data set for testing the effect of the variations in the shape, texture and orientation on the recognition performance. The results obtained in this work proved that the use of ANN for classification of plants based on the images of their leave is a promising

idea. It is proving the ability to use neural networks for leave recognition tasks and for machine vision use in the classification process.

REFERENCES

- [1] HARPER D., "Taxonomy". *Online Etymology Dictionary*. Retrieved April 18, 2011. Lecture notes, URL=< <http://speedyfacts101.webs.com/taxonomy.htm>>.
- [2] RUSSELL S. J., NORVIG P., *Artificial Intelligence: A Modern Approach*, 3rd ed 2009.
- [3] KASK K., (2015 July). *CompSci 271: Introduction to Artificial Intelligence, Fall 2014, lecture notes*,
URL=<<http://www.ics.uci.edu/~kkask/Fall2014 CS271/slides/01-intro-class.pdf>>.
- [4] Expert system. *Collins English Dictionary - Complete & Unabridged 10th Edition*. Retrieved July 26, 2015,
URL=<<http://dictionary.reference.com/browse/expert system>>.
- [5] GREGORY J. E. R., *Foundations of genetic algorithms*, URL=<
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html>.
- [6] ZADEH L. A., *Fuzzy logic and the calculus of fuzzy if-then rules*, in Proc. 22nd Intl. Symposium on Multiple-Valued Logic, Los Alamitos, IEEE Computer Society Press, 1992.
- [7] Wikipedia contributors, *Fuzzy logic*, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Fuzzy_logic&oldid=671385708 (accessed July, 2015).
- [8] SCHUMANN J., LIU Y., (Eds.), *Applications of Neural Networks in High Assurance Systems*, vol. 268, 2009.
- [9] STERGIO C., SIGANOS D., *Neural Networks*.
URL=<http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>.
- [10] NICI S., CUMMINS F., (2015 July), *Genevieve Orr: Neural networks, Fall 99, lecture notes*, URL=<<http://www.willamette.edu/~gorr/classes/cs449/intro.html>>.

- [11] JACOBSON L., (2014, March), *Introduction to Artificial Neural Networks Part 2 - Learning*, URL=<<http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-2-learning/8>>.
- [12] KADIR A., NUGROHO L.E., SUSANTO A., SANTOSA P.I., *Foliage Plant Identification*, International Journal of Computer Applications, Vol. 29, No 9 2011, 15-20.
- [13] AMLEKAR M., MANZA R.R., YANNAWAR P., GAIWAD A.T., *Leaf Features Based Plant Classification Using Artificial Neural Network*, IBMRD's Journal of Management and Research, Vol 3, No 1 2014,224-232.
- [14] ARIVAHAGAN S., NEWLIN S. R., ANANTHI S., VISHNU V.S., *Detection of unhealthy region of plant leaves and classification of plant leaf disease using texture features*. Agric Eng Int: CIGR Journal, Vol.15 , No 1 2013, 211-217.
- [15] KHAIRNAR K., DAGADE R., *Disease Detection and Diagnosis on Plant using Image Processing-A Review*, International Journal of Computer Applications, Vol 108, No 13 2014,36-38.
- [16] MANOJA M., RAJALAKSHMI J., *Early Detection of Pests on Leaves Using Support Vector Machine*, International Journal of Electronics Research Applications, Vol 2, No 4 2014, 187-193.
- [17] WU S. G., FORREST S. B., XU E. Y., WANG Y. X., CHANG Y. F., XIANG Q. L., *Leaf Recognition Algorithm For Plant Classification Using Probabilistic Neural Network*, IEEE International Symposium On Signal Processing And Information Technology, 2007.
- [18] SHITALA P., KRISHNA M. K., TRIPATHI R. C., *Relative sub-image based features for leaf recognition using support vector machine*. In Proceedings of the 2011 International Conference on Communication, Computing & Security (ICCCS '11), 2011, 343-346.
- [19] COPE J. S., REMAGNINO P., BARMAN S., WILKIN P., *Plant texture classification using gabor co-occurrences*,” in Proceedings of the 6th international conference on Advances in visual computing, 2, 2010, 669– 677.

[20] GERSHENSON C., *Artificial Neural Networks for Beginners*,
URL=<<http://arxiv.org/ftp/cs/papers/0308/0308031.pdf>>

Appendix A

Pictures of leaves used for noise experiments



Alligator_pear



Apple



Benjamin1



Benjamin2



Bougainvillea



Cherr_laurel



Duranta_tree



False_poplar



Gale



Carob



Israel_rubber



Kamkuat



Mandarinorange



New_world_fruit



Okuloptus



Oleander



Orange



Passion_fruit



Pitosporum



Solanum_rantonetti



Olive



Bottle_brush



Ficus



Laurus_nobilis



Lemon



Psidium



The_spindle_tree

Pictures of leaves used for Flavia database subset experiments



Pubescent
bamboo



Chinese horse
chestnut



Anhui Barberry



Pure indigo



Japanese maple



Japan Arrow
wood



Tangerine



Canadian poplar



Golden rain tree



Chinese cinnamon



Big-fruited
Holly



Japanese
cheesewood



Winter sweet



Camphortree



Ginkgo,
maidenhair tree



Crape myrtle,
Crepe myrtle



Oleander



Beale's barberry



Castor aralia



Ford Wood lotus

Appendix B

The following is the matlab code written for generating training and the testing images for the noise experiments.

```
%Creating noisy images for training and testing

clear all;
clc;
clear;

%list of trees

leafnames={
'alligator_pear.jpg'
'apple.jpg'
'benjamin1.jpg'
'benjamin2.jpg'
'bottle_brush.jpg'
'bougainvillea.jpg'
'cherry_laurel.jpg'
'duranta_tree.jpg'
'false_poplar.jpg'
'ficus.jpg'
'gale.jpg'
'harrup.jpg'
'israel_rubber.jpg'
'kamkuat.jpg'
'Laurus_nobilis.jpg'
'lemon.jpg'
'mandarinorange.jpg'
'new_world_fruit.jpg'
'okuloptus.jpg'
'oleander.jpg'
'orange.jpg'
'passion_fruit.jpg'
'pitosporum.jpg'
'Psidium.jpg'
'Solanum_rantonetti.jpg'
'the_spindle_tree.jpg'
'zeytin.jpg'
};

imgdir='/home/yucel/Downloads/Tezim/tez-matlab/50x50-gray/';
cd (imgdir);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                adding noise for training data                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%sp noise with 0.01
noisedir='sp-0.01';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'salt & pepper',0.01);
```

```

        noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
        imwrite(img,noisefname{1});
    end

%sp noise with 0.04
noisedir='sp-0.04';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'salt & pepper',0.04);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%sp noise with 0.08
noisedir='sp-0.08';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'salt & pepper',0.08);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%gaussian noise with 0.02 0.005
noisedir='gs-0.005';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'gaussian',0.02, 0.005);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%gaussian noise with 0.02 0.01
noisedir='gs-0.01';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'gaussian', 0.02, 0.01);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%speckle noise with 0.04
noisedir='s-0.04';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'speckle',0.04);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

```

```

%speckle noise with
noisedir='s-0.08';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'speckle',0.08);
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%poisson noise
noisedir='p';
mkdir(noisedir);
for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});
    img=imnoise(img,'poisson');
    noisefname=strcat(imgdir,noisedir,'/',leafnames(i));
    imwrite(img,noisefname{1});
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               adding noise for testing data                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imgdir='/home/yucel/Downloads/Tezim/tez-matlab/50x50-gray/';
cd (imgdir);

%Group A

%creating directories
noisedir='A'; mkdir(noisedir);

noisedir='A/1'; mkdir(noisedir);
noisedir='A/2'; mkdir(noisedir);
noisedir='A/3'; mkdir(noisedir);
noisedir='A/4'; mkdir(noisedir);

for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});

    img1=imnoise(img,'gaussian',0.03, 0.01);
    noisefname=strcat(imgdir,'A/1/',leafnames(i));
    imwrite(img1,noisefname{1});

    img2=imnoise(img,'salt & pepper',0.10);
    noisefname=strcat(imgdir,'A/2/',leafnames(i));
    imwrite(img2,noisefname{1});

    img3=imnoise(img,'speckle',0.12);
    noisefname=strcat(imgdir,'A/3/',leafnames(i));
    imwrite(img3,noisefname{1});

```

```

        img4=imnoise(img,'poisson');
        noisefname=strcat(imgdir,'A/4/',leafnames(i));
        imwrite(img4,noisefname{1});
    end

%Group B

%creating directories
noisedir='B'; mkdir(noisedir);

noisedir='B/1'; mkdir(noisedir);
noisedir='B/2'; mkdir(noisedir);
noisedir='B/3'; mkdir(noisedir);
noisedir='B/4'; mkdir(noisedir);

for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});

    img1=imnoise(img,'gaussian',0.03, 0.02);
    noisefname=strcat(imgdir,'B/1/',leafnames(i));
    imwrite(img1,noisefname{1});

    img2=imnoise(img,'salt & pepper',0.12);
    noisefname=strcat(imgdir,'B/2/',leafnames(i));
    imwrite(img2,noisefname{1});

    img3=imnoise(img,'speckle',0.22);
    noisefname=strcat(imgdir,'B/3/',leafnames(i));
    imwrite(img3,noisefname{1});

    img4=imnoise(img,'poisson');
    noisefname=strcat(imgdir,'B/4/',leafnames(i));
    imwrite(img4,noisefname{1});
end

%Group C

%creating directories
noisedir='C'; mkdir(noisedir);

noisedir='C/1'; mkdir(noisedir);
noisedir='C/2'; mkdir(noisedir);
noisedir='C/3'; mkdir(noisedir);
noisedir='C/4'; mkdir(noisedir);

for i=1:27
    fname=strcat(imgdir,leafnames(i));
    img=imread(fname{1});

    img1=imnoise(img,'gaussian',0.03, 0.03);
    noisefname=strcat(imgdir,'C/1/',leafnames(i));
    imwrite(img1,noisefname{1});

```

```

img2=imnoise(img,'salt & pepper',0.15);
noisefname=strcat(imgdir,'C/2/',leafnames(i));
imwrite(img2,noisefname{1});

img3=imnoise(img,'speckle',0.28);
noisefname=strcat(imgdir,'C/3/',leafnames(i));
imwrite(img3,noisefname{1});

img4=imnoise(img,'poisson');
noisefname=strcat(imgdir,'C/4/',leafnames(i));
imwrite(img4,noisefname{1});
end

```

The following is the matlab code written for generating feed forward neural network for the training and the testing of the images.

```

%Training and testing the ANN

```

```

clear all;
clc;
clear;

%list of trees

leafnames={
'alligator_pear.jpg'
'apple.jpg'
'benjamin1.jpg'
'benjamin2.jpg'
'bottle_brush.jpg'
'bougainvillea.jpg'
'cherry_laurel.jpg'
'duranta_tree.jpg'
'false_poplar.jpg'
'figus.jpg'
'gale.jpg'
'harrup.jpg'
'israel_rubber.jpg'
'kamkuat.jpg'
'Laurus_nobilis.jpg'
'lemon.jpg'
'mandarinorange.jpg'

```

```

'new_world_fruit.jpg'
'okuloptus.jpg'
'oleander.jpg'
'orange.jpg'
'passion_fruit.jpg'
'pitosporum.jpg'
'Psidium.jpg'
'Solanum_rantonetti.jpg'
'the_spindle_tree.jpg'
'zeytin.jpg'
};

imgdir='/home/yucel/Downloads/Tezim/tez-matlab/training-images/';
cd (imgdir);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 9 training directories each with 27 images %
% 243 samples of 50x50 images %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% gs-0.005
% gs-0.01
% p
% s-0.04
% s-0.08
% sp-0.01
% sp-0.04
% sp-0.08
% without-noise

traindirs={
'gs-0.005'
'gs-0.01'
'p'
's-0.04'
's-0.08'
'sp-0.01'
'sp-0.04'
'sp-0.08'
'without-noise'
};

```

```

%array of images
img_matrix = zeros(2500,243);
img_matrix_temp = zeros(2500,243);
img_data = zeros(2500,1);

%load train data
sample=1;
for j=1:9

    for i=1:27
        fname=strcat(imgdir,traindirs(j),'/',leafnames(i));

        img=imread(fname{1});

        x=1;
        for r=1:50
            for c=1:50
                img_data(x,1)=img(c,r);
                x=x+1;
            end
        end

        img_matrix_temp(:,sample)=img_data(:,1);
        sample=sample+1;

    end
end

%normalize inputs to [-1..1],

[img_matrix,PS] = mapminmax(img_matrix_temp,-1,1);

% output_vector size 27x243

```

```
output_vector=[eye(27) eye(27) eye(27) eye(27) eye(27) eye(27)
eye(27) eye(27) eye(27)];
```

```
%One hidden layer generally produces excellent results,
%but you may want to try two hidden layers, if the results with one
%are not adequate. Increasing the number of neurons in the hidden
%layer
%increases the power of the network,
%but requires more computation and is more likely to produce
%overfitting.
```

```
% training functions
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% trainlm Levenberg-Marquardt
% trainbr Bayesian Regularization
% trainbfg BFGS Quasi-Newton
% trainrp Resilient Backpropagation
% trainscg Scaled Conjugate Gradient
% traincgb Conjugate Gradient with Powell/Beale Restarts
% traincgf Fletcher-Powell Conjugate Gradient
% traincgp Polak-Ribière Conjugate Gradient
% trainoss One Step Secant
% traingdx Variable Learning Rate Gradient Descent
% traingdm Gradient Descent with Momentum
% traingd Gradient Descent
```

```
% As a note on terminology, the term "backpropagation" is sometimes
% used
```

```
% to refer specifically to the gradient descent algorithm,
% when applied to neural network training.
```

```
%
```

```
% Also, the multilayer network is sometimes referred to as a
% backpropagation network. However, the backpropagation technique
% that is used to compute gradients and Jacobians in a multilayer
% network can also be applied to many different network
% architectures.
```

```
%for a single hidden layers uncomment the line below
```

```

%net = feedforwardnet(240,'traingdx');

%for 2 hidden layers uncomment the line below
net = feedforwardnet([200 240],'traingdx');

%net = patternnet([200 240],'traingdx');

% Feedforward networks often have one or more hidden layers of
%sigmoid
% neurons followed by an output layer of linear neurons. Multiple
% layers of neurons with nonlinear transfer functions allow the
% network to learn nonlinear relationships between input and output
% vectors. The linear output layer is most often used for function
% fitting (or nonlinear regression) problems.
% On the other hand, if you want to constrain the outputs of a
%network
% (such as between 0 and 1), then the output layer should use a
% sigmoid transfer function (such as logsig). This is the case
% when the network is used for pattern recognition problems
% (in which a decision is being made by the network).
%

% Transfer function of the hidden layers. incase single layer use
%only
% net.layers{1}.transferFcn

net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';

% use logsig for output layer since values are 0 or 1
% in case of a single hidden layer layers{2} is the output layer
% in case of 2 hidden layers layers{3} is the output layer
% remember logsig(0) gives 0.5!!!
% for this we must use purelin and also input must be normalized to
%[-1..1]

net.layers{3}.transferFcn = 'purelin';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Training parameters %

```

```

% with default values %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% net.trainParam.epochs    1000 Maximum number of epochs to train
% net.trainParam.goal      0 Performance goal
% net.trainParam.showCommandLine false Generate comm-line output
% net.trainParam.showWindow      true      Show training GUI
% net.trainParam.lr           0.01      Learning rate
% net.trainParam.max_fail     6          Maximum validation failures
% net.trainParam.min_grad     1e-5 Minimum performance gradient
% net.trainParam.show 25 Epochs between disp (NaN for no displays) %
% net.trainParam.time  inf      Maximum time to train in seconds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Training parameters specific to traingdx with default values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% net.trainParam.lr_inc      1.05 Ratio to increase learning rate
% net.trainParam.lr_dec      0.7  Ratio to decrease learning rate
% net.trainParam.max_fail    6     Maximum validation failures
% net.trainParam.max_perf_inc 1.04 Maximum performance increase
% net.trainParam.mc          0.9   Momentum constant
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Learning rate default 0.01
net.trainParam.lr = 0.05;

% Error tolerance, stopping criterion
net.trainParam.goal = 1e-4;

% Minimum performance gradient
net.trainParam.min_grad = 1e-5;

% Maximum number of epochs to train
net.trainParam.epochs = 1500;

net = configure(net,img_matrix,output_vector);

% do not divide input data for validation and testing

```

```

net.divideFcn = '';

% trTraining record (epoch and perf)

net=init(net);
[net,tr] = train(net,img_matrix,output_vector);
view(net)

y = net(img_matrix);

perf = perform(net,y,output_vector);

%to save trained network to a file uncomment the following line
save('/home/yucel/Downloads/Tezim/tez-matlab/training.mat','net');

%testing with original train input
outputs = sim(net,img_matrix) ;
dlmwrite('/home/yucel/Downloads/Tezim/tez-matlab/testing-train-
data.txt',outputs,'delimiter','\t','precision','%.2f');

% For windows
%xlswrite('/home/yucel/Downloads/Tezim/tez-matlab/testing-train-
%data.xlsx',outputs);

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile =fopen('/home/yucel/Downloads/Tezim/tez-matlab/testing-train-
data-idx.txt','wt');
match=0;

for i=1:n(2)
    check=mod(i,27);
    if check == 0
        check=27;
    end
    if idx_of_max(i) == check
        match = match + 1;
    end
    fprintf(myfile,'%d %.3f %d\n',check, maxout(i),idx_of_max(i));

```

```

end

fprintf(myfile, 'Recognition rate : %.3f %d of %d\n', ((100 * match) /
n(2)), match, n(2));
fclose(myfile);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Testing 3 Groups I, II, III                               %
%                               Each group has 4 noise type                               %
%                               4x27 = 108 samples of 50x50 testing images               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

uiwait(helpdlg('Training finished! Continue with testing!'));

testdir='/home/yucel/Downloads/Tezim/tez-matlab/testing-images/';

test_matrix = zeros(2500,108);
test_matrix_temp = zeros(2500,108);
img_data = zeros(2500,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Testing for Level I                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First load image data from files and feed to network
sample=1;
for i=1:27
    noisefname=strcat(testdir, 'A/1/', leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir, 'A/2/', leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

```

```

    noisefname=strcat(testdir,'A/3/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'A/4/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

end

%normalize inputs to [-1..1],

[test_matrix,PS] = mapminmax(test_matrix_temp,-1,1);

%testing with images from group A
outputs = sim(net,test_matrix) ;
dlmwrite('/home/yucel/Downloads/Tezim/tez-
matlab/testA.txt',outputs,'delimiter','\t','precision','%.2f');

% lets find the max values of the columns of the output vector
% and the indices of the max values
% where indices are the image numbers
% since we test 4 instances of the same images but different noises
% the ideal output vector should contain 4 1s in the form
% of the descending scale!

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile=fopen('/home/yucel/Downloads/Tezim/tez-matlab/testAmax-
idx.txt','wt');
match=0;

for i=1:n(2)
    if idx_of_max(i) == ceil(i/4)
        match = match + 1;
    end
    fprintf(myfile,'%d %.3f %d\n',ceil(i/4), maxout(i),idx_of_max(i));

```

```

end

fprintf(myfile,'Recognition rate : %.3f %d of %d\n',((100 * match) /
n(2)), match, n(2));
fclose(myfile);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Testing for Level II                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First load image data from files and feed to network

sample=1;

for i=1:27

    noisefname=strcat(testdir,'B/1/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'B/2/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'B/3/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'B/4/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

```

```

end

%normalize inputs to [-1..1],

[test_matrix,PS] = mapminmax(test_matrix_temp,-1,1);

%testing with images from group B
outputs = sim(net,test_matrix) ;
dlmwrite('/home/yucel/Downloads/Tezim/tez-
matlab/testB.txt',outputs,'delimiter','\t','precision','%.2f');

% lets find the max values of the columns of the output vector
% and the indices of the max values
% where indices are the image numbers
% since we test 4 instances of the same images but different noises
% the ideal output vector should contain 4 1s in the form
% of the descending scale!

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile      =fopen('/home/yucel/Downloads/Tezim/tez-matlab/testBmax-
idx.txt','wt');
match=0;

for i=1:n(2)
    if idx_of_max(i) == ceil(i/4)
        match = match + 1;
    end
    fprintf(myfile,'%d %.3f %d\n',ceil(i/4), maxout(i),idx_of_max(i));
end

fprintf(myfile,'Recognition rate : %.3f %d of %d\n',((100 * match) /
n(2)), match, n(2));
fclose(myfile);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Testing for Level III           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sample=1;
for i=1:27

    noisefname=strcat(testdir,'C/1/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'C/2/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'C/3/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

    noisefname=strcat(testdir,'C/4/',leafnames(i));
    img=imread(noisefname{1});
    img_data(:,1)=img(:);
    test_matrix_temp(:,sample)=img_data(:,1);
    sample=sample+1;

end

%normalize inputs to [-1..1],

[test_matrix,PS] = mapminmax(test_matrix_temp,-1,1);

%testing with images from group B
outputs = sim(net,test_matrix) ;
dlmwrite('/home/yucel/Downloads/Tezim/tez-
matlab/testC.txt',outputs,'delimiter','\t','precision','%2f');

```

```

% lets find the max values of the columns of the output vector
% and the indices of the max values
% where indices are the image numbers
% since we test 4 instances of the same images but different noises
% the ideal output vector should contain 4 1s in the form
% of the descending scale!

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile      =fopen('/home/yucel/Downloads/Tezim/tez-matlab/testCmax-
idx.txt','wt');
match=0;

for i=1:n(2)
    if idx_of_max(i) == ceil(i/4)
        match = match + 1;
    end
    fprintf(myfile,'%d %.3f %d\n',ceil(i/4), maxout(i),idx_of_max(i));

end

fprintf(myfile,'Recognition rate : %.3f %d of %d\n',((100 * match) /
n(2)), match, n(2));
fclose(myfile);

```

The following is the matlab code written for generating training and the testing images for the Flavia database subset experiments.

```
%Training and testing the ANN
clear all; clc; clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Here you set parameters!!!
tree_no = 20; %number of trees
leaf_per_tree = 10; %leaf per tree
train_no = 7; %number of leaves for training
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % for yaprak8-gray-50x50/
% % 20 trees
% % 10 copies of each leaf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rootdir='/home/yucel/Downloads/yaprak/yaprak8-gray-50x50/';
leafdirs={ 'directory1'
            'directory2'
            'directory3'
            'directory4'
            'directory5'
            'directory6'
            'directory7'
            'directory8'
            'directory9'
            'directory10'
            'directory11'
            'directory12'
            'directory13'
            'directory14'
            'directory15'
            'directory16'
            'directory17'
            'directory18'
            'directory19'
            'directory20'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%next these parameters will be calculated
test_no = leaf_per_tree - train_no;
%number of leaves for testing

train_leaf_no = tree_no * train_no;
%total number of leaves for training

test_leaf_no = tree_no * test_no;
%total number of leaves for testing


fprintf('We have %d different trees.\n', tree_no);
fprintf('We have %d different leaves per tree.\n', leaf_per_tree);
fprintf('For train using %d leaves of each tree.\n', train_no);
fprintf('For train using %d samples.\n', train_leaf_no);


fprintf('For testing using %d leaves of each tree.\n', test_no);
fprintf('For testing using %d samples.\n', test_leaf_no);
fprintf('press any key\n');
pause;


%array of images
train_matrix = zeros(2500,train_leaf_no);
train_matrix_temp = zeros(2500,train_leaf_no);
temp_img_data = zeros(2500,1);
test_matrix = zeros(2500,test_leaf_no);
test_matrix_temp = zeros(2500,test_leaf_no);


train_sample=1;
test_sample=1;
for j=1:tree_no
    %enter tree dir
    treedirname=strcat(rootdir,leafdirs{j},'/');
    %cd(treedirname);
    fprintf('%s\n', treedirname);
    %read leaves to matrix
    %load train array
    for i=0:train_no-1
        fname=strcat(treedirname,num2str(i),'.jpg');
        fprintf('train %d: %s\n', train_sample,fname);
        img=imread(fname);
    end
end

```

```

        x=1;
        for r=1:50
            for c=1:50
                temp_img_data(x,1)=img(c,r);
                x=x+1;
            end
        end
        train_matrix_temp(:,train_sample)= temp_img_data(:,1);
        train_sample=train_sample+1;
    end

    %load test array
    for i=train_no:leaf_per_tree-1
        fname=strcat(treedirname,num2str(i),'.jpg');
        fprintf('test %d: %s\n', test_sample, fname);
        img=imread(fname);
        x=1;
        for r=1:50
            for c=1:50
                temp_img_data(x,1)=img(c,r);
                x=x+1;
            end
        end
        test_matrix_temp(:,test_sample) = temp_img_data(:,1);
        test_sample=test_sample+1;
    end
end

%normalize inputs to [-1..1], for training
%mapminmax works on rows!!!!
%for this reason work on transpose matrix!!!!
[transpose_train_matrix,PS] = mapminmax(train_matrix_temp',-1,1);
train_matrix = transpose_train_matrix';

%normalize inputs to [-1..1], for testing

[transpose_test_matrix,PS] = mapminmax(test_matrix_temp',-1,1);

test_matrix = transpose_test_matrix';

```

```

output_vector=zeros(tree_no , tree_no * train_no);

%first -1 everywhere
for i=1:tree_no
    for j=1:(tree_no * train_no)
        output_vector(i, j)= -1;
    end
end

%then put 1s for the train image like a steps of a scale
row = 1;
ones = 0;
for j=1:(tree_no * train_no)
    if (ones ~= train_no)
        output_vector(row, j)= 1;
        ones = ones + 1;
    else
        row = row + 1;
        output_vector(row, j)= 1;
        ones = 1;
    end
end

%One hidden layer generally produces excellent results,
%but you may want to try two hidden layers, if the results with one
%are not adequate. Increasing the number of neurons in the hidden
%layer
%increases the power of the network,
%but requires more computation and is more likely to produce
%overfitting.

% training functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% trainlm  Levenberg-Marquardt
% trainbr  Bayesian Regularization
% trainbfg BFGS Quasi-Newton
% trainrp  Resilient Backpropagation
% trainscg Scaled Conjugate Gradient
% traincgb Conjugate Gradient with Powell/Beale Restarts
% traincgf Fletcher-Powell Conjugate Gradient

```

```

% traincgp Polak-Ribière Conjugate Gradient
% trainoss One Step Secant
% traingdx Variable Learning Rate Gradient Descent
% traingdm Gradient Descent with Momentum
% traingd Gradient Descent

% As a note on terminology, the term "backpropagation" is sometimes
%used
% to refer specifically to the gradient descent algorithm,
% when applied to neural network training.
% Also, the multilayer network is sometimes referred to as a
% backpropagation network. However, the backpropagation technique
% that is used to compute gradients and Jacobians in a multilayer
% network can also be applied to many different network
%architectures.

%for a single hidden layers uncomment the line below
%net = feedforwardnet(300,'traingdx');

%for 2 hidden layers uncomment the line below
net = feedforwardnet([200 240],'traingdx');

% Feedforward networks often have one or more hidden layers of %
%sigmoid
% neurons followed by an output layer of linear neurons. Multiple
% layers of neurons with nonlinear transfer functions allow the
% network to learn nonlinear relationships between input and output
% vectors. The linear output layer is most often used for function
% fitting (or nonlinear regression) problems.
% On the other hand, if you want to constrain the outputs of a %
%network
% (such as between 0 and 1), then the output layer should use a
% sigmoid transfer function (such as logsig). This is the case
% when the network is used for pattern recognition problems
% (in which a decision is being made by the network).
% Transfer function of the hidden layers.
% incase single layer use only
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';

% use logsig for output layer since values are 0 or 1

```

```

% in case of a single hidden layer layers{2} is the output layer
% in case of 2 hidden layers layers{3} is the output layer
% remember logsig(0) gives 0.5!!!
% for this we must use purelin and also input must be normalized to
%[-1..1]

net.layers{3}.transferFcn = 'purelin';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Training parameters with default values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% net.trainParam.epochs 1000 Maximum number of epochs to train
% net.trainParam.goal 0 Performance goal
% net.trainParam.showCommandLine false Generate comm-line output
% net.trainParam.showWindow true Show training GUI
% net.trainParam.lr 0.01 Learning rate
% net.trainParam.max_fail 6 Maximum validation failures
% net.trainParam.min_grad 1e-5 Minimum performance gradient
% net.trainParam.show 25 Epochs between displays (NaN for no
%displays)
% net.trainParam.time inf Maximum time to train in seconds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Training parameters specific to traingdx with default values %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% net.trainParam.lr_inc 1.05 Ratio to increase learning rate
% net.trainParam.lr_dec 0.7 Ratio to decrease learning rate
% net.trainParam.max_fail 6 Maximum validation failures
% net.trainParam.max_perf_inc 1.04 Maximum performance increase
% net.trainParam.mc 0.9 Momentum constant
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% default 0.9 Momentum constant
% net.trainParam.mc = 0.85;

% Learning rate default 0.01
net.trainParam.lr = 0.05;

% Error tolerance, stopping criterion
net.trainParam.goal = 1e-7;

% Minimum performance gradient
net.trainParam.min_grad = 1e-6;

```

```

% Maximum number of epochs to train
net.trainParam.epochs = 2000;

net = configure(net,train_matrix,output_vector);

% do not divide input data for validation and testing
net.divideFcn = '';

% trTraining record (epoch and perf)
net=init(net);
[net,tr] = train(net,train_matrix,output_vector);
view(net)

y = net(train_matrix);

perf = perform(net,y,output_vector);

%to save trained network to a file uncomment the following line
save('/home/yucel/Downloads/yaprak/training.mat','net');

%testing with original train input
outputs = sim(net,train_matrix) ;
dlmwrite('/home/yucel/Downloads/yaprak/testing-train-
data.txt',outputs,'delimiter','\t','precision','%2f');

% For windows
% xlswrite('/home/yucel/Downloads/Tezim/tez-matlab/testing-train-
data.xlsx',outputs);

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile =fopen('/home/yucel/Downloads/yaprak/testing-train-data-
idx.txt','wt');
match=0;

for i=1:n(2)
    check=ceil(i/train_no);
    if idx_of_max(i) == check
        match = match + 1;
    end
end

```

```

fprintf(myfile,'%d %.3f %d\n',check, maxout(i),idx_of_max(i));

end

fprintf(myfile,'Recognition rate : %.3f %d of %d\n',((100 * match) /
n(2)), match, n(2));
fclose(myfile);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Testing                               %
%      test_no x tree_no samples of 50x50 testing images          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

uiwait(helpdlg('Training finished! Continue with testing!'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Testing                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%testing with images

% to test the process use train_matrix and uncomment the following
%line for
% debugging puposes!!!
% outputs = sim(net,train_matrix) ;

outputs = sim(net,test_matrix) ;
dlmwrite('/home/yucel/Downloads/yaprak/test.txt',outputs,'delimiter'
,'\t','precision','%.2f');

% lets find the max values of the columns of the output vector
% and the indices of the max values
% where indices are the image numbers
% since we test 4 instances of the same images but different noises
% the ideal output vector should contain 4 1s in the form
% of the descending scale!

[maxout,idx_of_max]=max(outputs);
n=size(maxout);
myfile = fopen('/home/yucel/Downloads/yaprak/testmax-idx.txt','wt');
match=0;

```

```

for i=1:n(2)
    if idx_of_max(i) == ceil(i/test_no)
        match = match + 1;
    end
    fprintf(myfile,'%d %.3f %d\n',ceil(i/test_no),
maxout(i),idx_of_max(i));
end

fprintf(myfile,'Recognition rate : %.3f %d of %d\n',((100 * match) /
n(2)), match, n(2));
fclose(myfile);

```