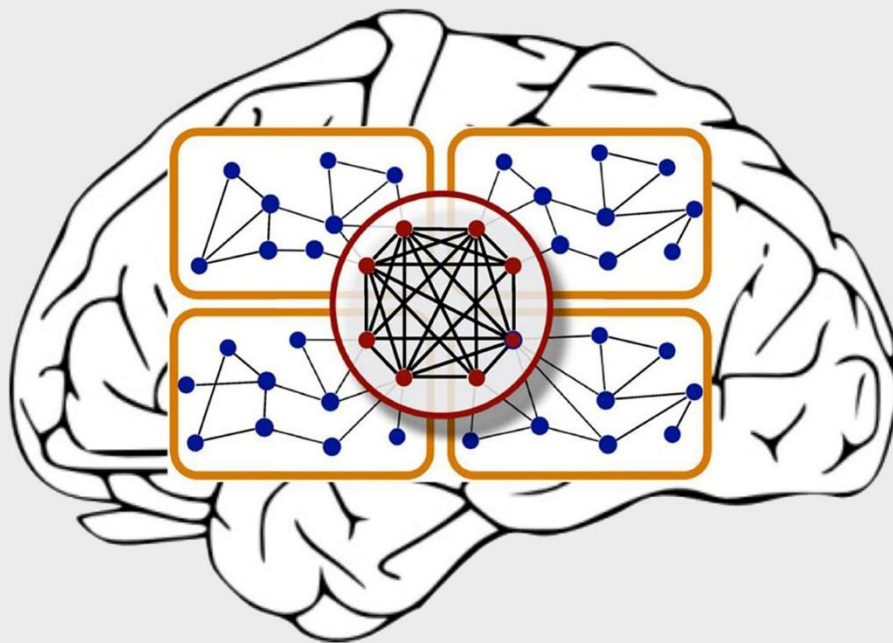


UNIVERSITY OF TRIPOLI

**ELECTRICAL AND ELECTRONIC
ENGINEERING DEPARTMENT**

B.SC.Project

***Data recognition
using Digital Image Processing
with Artificial Neural Network***



**Prepared by
Murad Mohammed AlBarki**

**Supervised by
Dr. Ali Elmelhi**

Tripoli - Libya

Spring 2018

Acknowledgement

I would like to express my special thanks and gratitude to my advisor Dr. Ali Elmelhi who gave me the golden opportunity to do this project. I would also like to thank all other faculties, laboratory and administrative staff of Department of Electrical and Electronic Engineering for their help. And special thanks to my family and friends for their support to finish this project.

Abstract

We live in technology generation where we try to connect everything to computer and use it either to solve our problems or to give us comfort by doing the work on behalf of us, and to relate the computer to our life we have to use tools and methods to represent some of our abilities, such as images translation, thinking and choosing. For image translation we use Digital Image Processing which is the use of computer algorithms to perform processes on digital images in order to get an enhanced image or to extract some useful information from it, and Artificial Neural Network works on simulating neurons in the brain, so the computer can think and choose, the main applications of Neural Network have been originally applied in control system engineering, this is due to the ability of training process for the Neural Networks where weights and biases are adjusted according to the error difference between the target and the actual output of the considered Neural Network.

Data recognition from images can help us in various fields, from taking the attendance to reading signs in foreign countries. In this project we mainly concerned in coding and designing a program using MATLAB software to simulate the backpropagation algorithm for training the feed-forward Artificial Neural Network for data recognition such as (Words, Numbers and Signatures) from given images, which can be achieved with the contribution of some certain digital image processes, and in which convert the image to a vector contains relative error of black pixels to white pixels in every column of pixels in the image, then use this vector as a training input of the Neural Network.

الملخص

نحن نعيش في جيل التكنولوجيا حيث نحاول توصيل كل شيء بالكمبيوتر واستخدامه إما لحل مشاكلنا أو لإعطائنا الراحة من خلال القيام بالعمل نيابة عنا، ولربط الكمبيوتر بحياتنا علينا استخدام أدوات وطرق لتقوم مقام بعض من قدراتنا، مثل ترجمة الصور، التفكير والاختيار. بالنسبة إلى ترجمة الصور، يتم استخدام معالجة الصور الرقمية التي تستخدم خوارزميات الكمبيوتر لإجراء عمليات على الصور الرقمية وذلك من أجل الحصول على صورة محسنة أو لاستخراج بعض المعلومات المفيدة منها، وتعمل الشبكة العصبية الاصطناعية على محاكاة الخلايا العصبية في الدماغ، ليتمكن الكمبيوتر من التفكير والاختيار، تم تطبيق التطبيقات الأساسية للشبكات العصبية في هندسة أنظمة التحكم، وذلك يرجع إلى إمكانية إجراء عملية تدريب الشبكة العصبية، حيث يتم تعديل الأوزان والتحيزات بناء على فرق الخطأ بين الهدف والخرج الفعلي للشبكة العصبية.

يمكن أن يساعدنا التعرف على البيانات من الصور في مختلف المجالات، بدءًا من أخذ الحضور إلى قراءة العلامات في البلدان الأجنبية. في هذا المشروع، يتركز العمل بشكل أساسي على برمجة وتصميم برنامج بإستعمال برمجية الماتلاب، لمحاكات خوارزمية الانتشار العكسي لتدريب الشبكات العصبية الصناعية ذات التغذية الأمامية، وذلك لغرض تمييز معلومات مثل (كلمات، أرقام، مواقع) من الصور المعطية، والذي يتم تحقيقه بمساهمة بعض عمليات معالجة الصور الرقمية المحددة، التي تحول الصورة الى متجه يحتوي على الخطأ النسبي للنقاط الضوئية (البكسلات) السوداء الى البيضاء في كل عمود من نقاط الصورة، ومن ثم استخدام هذا المتجه كدخل تدريبي للشبكة العصبية.

Contents

List of Figures	VI
Chapter 1 Introduction	1
1.1 General Overview	2
1.1.1 Artificial Neural Networks (ANN)	2
1.1.2 Digital Image Processing	5
1.2 Project Objective	5
1.3 Project Outline	5
Chapter 2 Artificial Neural Networks	7
2.1 Introduction	8
2.2 Neural Network Structure	9
2.2.1 Simple Neuron	10
2.2.2 A more Complicated Neuron	11
2.3 Architecture of Neural Network	12
2.3.1 Feed-Forward Networks	12
2.3.2 Feedback Networks	13
2.4 Learning Process Paradigm	14
2.5 Machine Learning Algorithms	15
2.6 Transfer Functions (Activation Functions)	17
2.7 Applications of Neural Networks	18
2.7.1 Neural Networks in Medicine	19
2.7.2 Neural Networks in Business	19
2.7.3 Neural Networks in Marketing	20
2.7.4 Neural Networks in Control Systems	20
2.8 Advantages of Neural Networks	22
2.9 Neural Networks versus Conventional Computers	22

Chapter 3	Backpropagation	25
3.1	Introduction	26
3.2	Gradient Descent	27
3.3	Backpropagation Requirements	28
3.4	Training Steps	29
Chapter 4	Digital Image Processing	41
4.1	Introduction	42
4.2	Digital Image Processing Algorithms Usages	43
4.3	Image Processing Techniques	44
4.3.1	Image Preprocessing	44
4.3.2	Image Segmentation	48
4.3.3	Feature Extraction	49
4.3.4	Image Classification	50
4.4	Image Processing Applications	51
Chapter 5	Software Simulation	53
5.1	Introduction	54
5.1.1	MATLAB GUI using GUIDE	54
5.2	Digital Image Processing Functions	56
5.3	Project Flowchart	60
5.4	Simulation & Results	60
5.5	Applications for this Project	76
Conclusion		78
Future Work		79
References		80
Appendix		83

List of Figures

Figure 1.1: a) Components of a neuron. b) The synapse	3
Figure 1.2: Simple neural network model diagram	4
Figure 2.1: Neural Network layers	9
Figure 2.2: Simple neuron	10
Figure 2.3: MCP neuron	11
Figure 2.4: MCP mathematically simulation	12
Figure 2.5: A simple feedforward network	13
Figure 2.6: A complicated network	14
Figure 3.1: Gradient Descent example	27
Figure 3.2: Training algorithm	29
Figure 3.3: Optimization operation	33
Figure 3.4: Weights calculating schema using Backpropagation	34
Figure 3.5: Using of chain rule	35
Figure 4.1: Counting circular objects in an image	44
Figure 4.2: Contrast stretching	46
Figure 4.3: Noise Removal	47
Figure 4.4: Image Classification	51
Figure 5.1: The GUIDE interface	55
Figure 5.2: Convert RGB image to grayscale	56
Figure 5.3: Convert grayscale image to binary image (black and white)	57
Figure 5.4: Finding the edges	57
Figure 5.5: Dilating the Image	58
Figure 5.6: IMFILL operation	58

Figure 5.7: Label matrix	59
Figure 5.8: Cropping operation	59
Figure 5.9: Project flowchart	60
Figure 5.10: Project main interface	61
Figure 5.11: Samples Image representations	62
Figure 5.12: EDGE effect	63
Figure 5.13: Dilated image	63
Figure 5.14: Filled dilated image	63
Figure 5.15: Labeling matrix for a binary image of 3 shapes	64
Figure 5.16: Cropping samples from an image	64
Figure 5.17: Cropping the edges	65
Figure 5.18: Extracting errors vector from a resized image	66
Figure 5.19: Collecting error vectors in one matrix	66
Figure 5.20: Created feed-forward neural network in MATLAB	66
Figure 5.21: Project Neural Network structure	67
Figure 5.22: Neural Network training tool interface	68
Figure 5.23: Learning rate over the epochs	69
Figure 5.24: Gradient result	69
Figure 5.25: performance result	70
Figure 5.26: Regression result	70
Figure 5.27: Adjusted weights of the hidden layer	71
Figure 5.28: Adjusted weights of the output layer	71
Figure 5.29: Biases values for each neuron in the three layers	71
Figure 5.30: testing image	72
Figure 5.31: Convert RGB testing image to grayscale	72
Figure 5.32: Convert the grayscale of testing image to black and white image	73

Figure 5.33: Finding the edges of black and white image	73
Figure 5.34: Dilating the black and white Image	73
Figure 5.35: Filling dilated image regions and holes	74
Figure 5.36: Finding the limits of the shapes	74
Figure 5.37: The output layer at testing one of the shapes	75
Figure 5.38: Arabic handwritten words recognition	76
Figure 5.39: Signatures recognition	77
Figure 5.40: Vehicles registration plate recognition	77

Chapter 1

Introduction

1.1 General Overview

Over the past two decades and as a result of the Artificial Intelligence and Robotics revolution, Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever-increasing amounts of data becoming available there is a good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. [1]

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. [2]

1.1.1 Artificial Neural Networks (ANN)

ANN are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural Networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize. [3]

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, as shown in (Figure 1.1a), a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches. At the end of each branch, and as shown in (Figure 1.1b) a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. [3]

Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. [3]

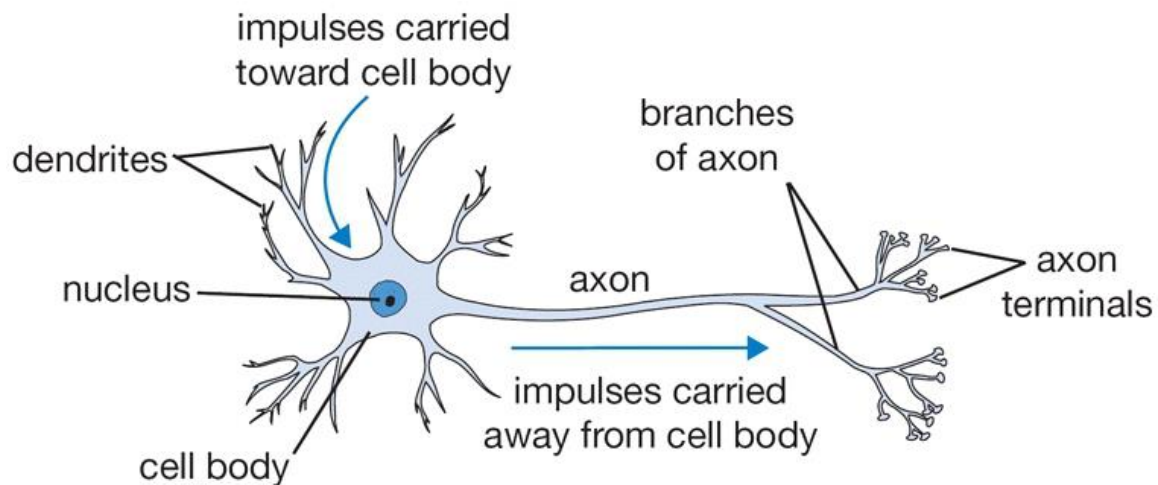


Figure 1.1a: Components of a neuron

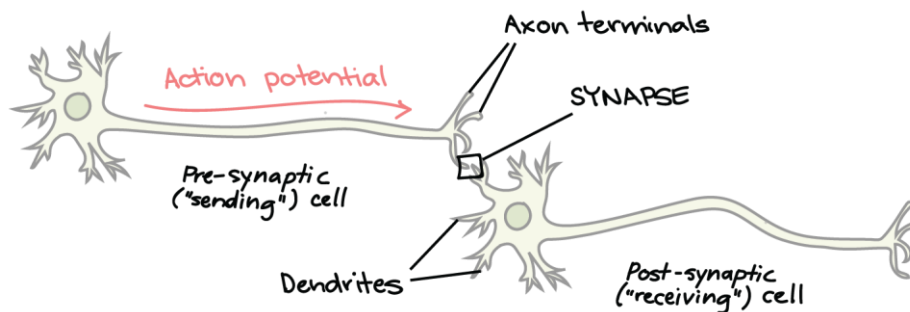


Figure 1.1b: The synapse

In computers neural networks usually involves a large number of processors operating in parallel and arranged in layers. The first layer receives the raw input information, analogous to optic nerves in human visual processing. Each successive layer receives the output from the layer preceding it, rather than from

the raw input, in the same way neurons further from the optic nerve receive signals from those closer to it. The last layer produces the output of the system.

Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted connections. The hidden layers then link to an 'output layer' where the answer is the output. (Figure 1.2) contains simple neural network model diagram.

Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs. [4]

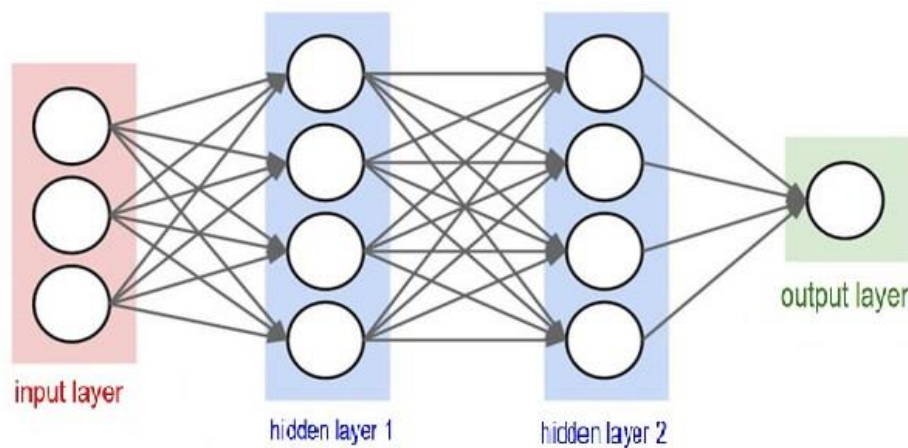


Figure 1.2: Simple neural network model diagram

1.1.2 Digital Image Processing

In computer science, digital image processing is the use of computer algorithms on digital images, a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics (features) associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too. [5]

1.2 Project Objective

The objects of this project are:

- Understand and explain the Artificial Neural Networks structure.
- Demonstrate back-propagation method in training Neural Networks.
- Present the principle of Digital Image Processing.
- Simulate the training process of Neural Networks using back-propagation method, and then simulate this network to recognize the data from considered images, after implementing some certain digital image processes on them.

1.3 Project Outline

Chapter 2 will discuss the Artificial neural networks (ANNs) structure in more details, explains its mechanism and overviews the most famous training algorithms and methods used in ANNs.

Chapter 3 will study the backpropagation method, which had been used as a training method in this project, then Chapter 4 will explain the concept of digital Image processing.

The first section of Chapter5 (Software Simulation) will talk about the software used in this project MATLAB, and Graphical User Interface (GUI), then thereafter section will discuss the simulation of the Neural Network, training and results that have been achieved to serve the purpose of the project.

Finally, the conclusion and future work will be included at the end of this project.

Chapter 2

Artificial Neural Networks

2.1 Introduction

The simplest definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen. He defined a neural network as:

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

ANNs are processing systems (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales. A large ANN might have hundreds or thousands of processor units, whereas a mammalian brain has billions of neurons with a corresponding increase in magnitude of their overall interaction and emergent behavior. Although ANN researchers are generally not concerned with whether their networks accurately resemble biological systems, some have. For example, researchers have accurately simulated the function of the retina and modeled the eye rather well. [6]

Although the mathematics involved with neural networking is not a trivial matter, a user can rather easily gain at least an operational understanding of their structure and function.

A neural network is an artificial-intelligence processing method within a computer that allows self-learning from experience. Neural nets can develop conclusions from a complex and seemingly unrelated set of information, it's configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. [6]

2.2 Neural Network Structure

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. The layer is 'input layer', which communicates to one or more 'hidden layers'. The hidden layers then link to an 'output layer' as shown in (Figure 2.1) below.

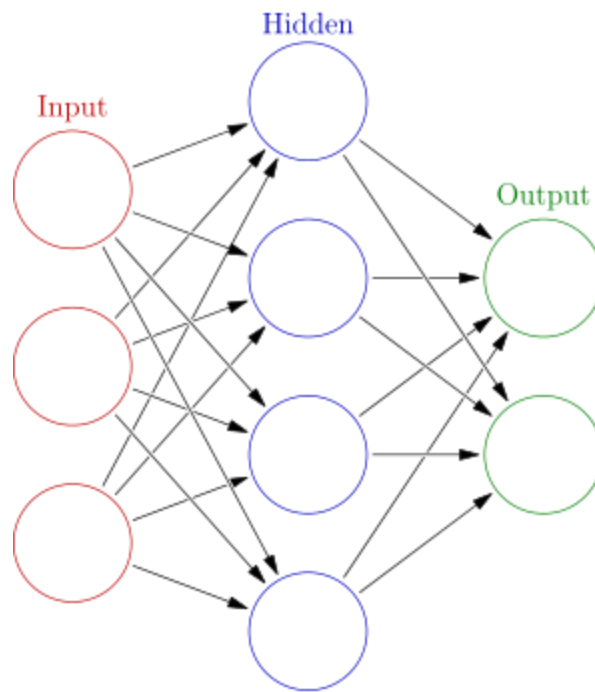


Figure 2.1: Neural Network layers

Neural networks are sometimes described in terms of their depth, including how many layers they have between input and output, or the model's so-called hidden layers. They can also be described by the number of hidden nodes the model has or in terms of how many inputs and outputs each node has. Variations on the classic neural-network design allow various forms of forward and backward propagation of information among layers.

Each processing node has its own small sphere of knowledge, including what it has seen and any rules it was originally programmed with or developed for itself. The layers are highly interconnected, which means each node in layer n will be connected to many nodes in layer $n-1$ (its inputs) and in layer $n+1$, which provides input for those nodes. There may be one or multiple nodes in the output layer, from which the answer it produces can be read.

Neural networks are notable for being adaptive, which means they modify themselves as they learn from initial training and subsequent runs provide more information about the world. The most basic learning model is centered on weighting the input streams, which is how each node weights the importance of input from each of its predecessors. Inputs that contribute to getting right answers are weighted higher. [6]

2.2.1 Simple Neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not, (Figure 2.2) shows the simple neuron where ($X_n, n = 1, 2, \dots, n$) are the inputs when the neuron at using mode. [7]

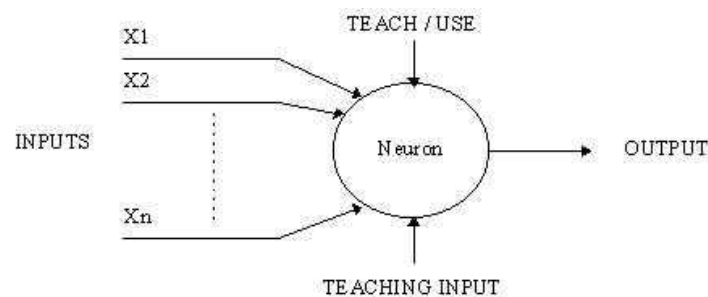


Figure 2.2: Simple neuron

2.2.2 A more Complicated Neuron

The previous neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron as shown in (Figure 2.3) is the McCulloch and Pitts model (MCP), where $(X_n, n = 1, 2, \dots, n)$ are the inputs when the neuron is at using mode, and $(W_n, n = 1, 2, \dots, n)$ are the adjustable weights. The difference from the previous model is that the inputs are 'weighted', the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire. [7]

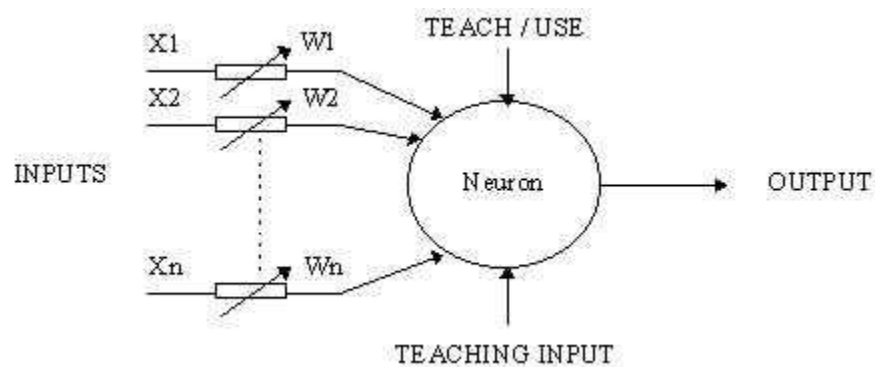


Figure 2.3: MCP neuron

In mathematical terms, the neuron fires if and only if;

$$X_1W_1 + X_2W_2 + X_3W_3 + \dots > T$$

Where T is the pre-set threshold value, (Figure 2.4) explain how to simulate MCP mathematically.

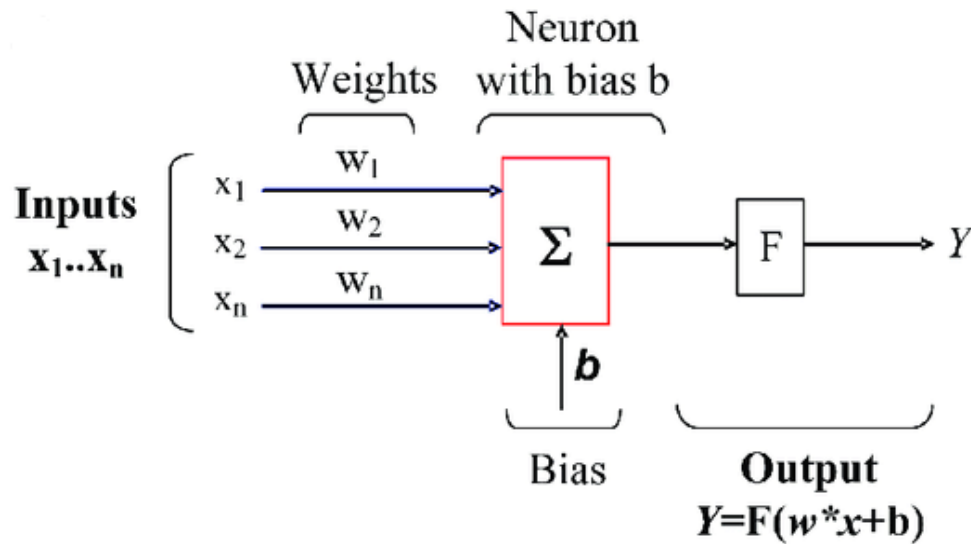


Figure 2.4: MCP mathematically simulation

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold.

Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back-error propagation. [7]

2.3 Architecture of Neural Networks

2.3.1 Feed-forward Networks

Feed-forward ANNs allow signals to travel one way only; from input to output, as (Figure 2.5) shows. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

Pattern Recognition is an important application of neural networks. It can be implemented by using a feed-forward neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern. [8]

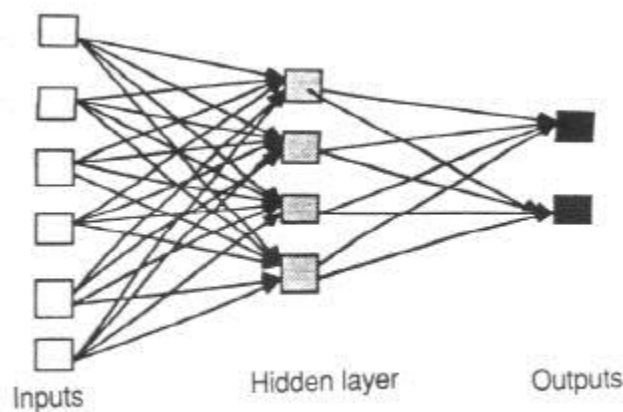


Figure 2.5: A simple feedforward network

2.3.2 Feedback Networks

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. (Figure 2.6) shows a feedback network, where $(U_n, n = 1, 2, \dots, n)$ are the neurons and $W_{i,j}$ ($i = 1, 2, \dots, n$), ($j = 1, 2, \dots, n$) are the weights between them. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.[8]

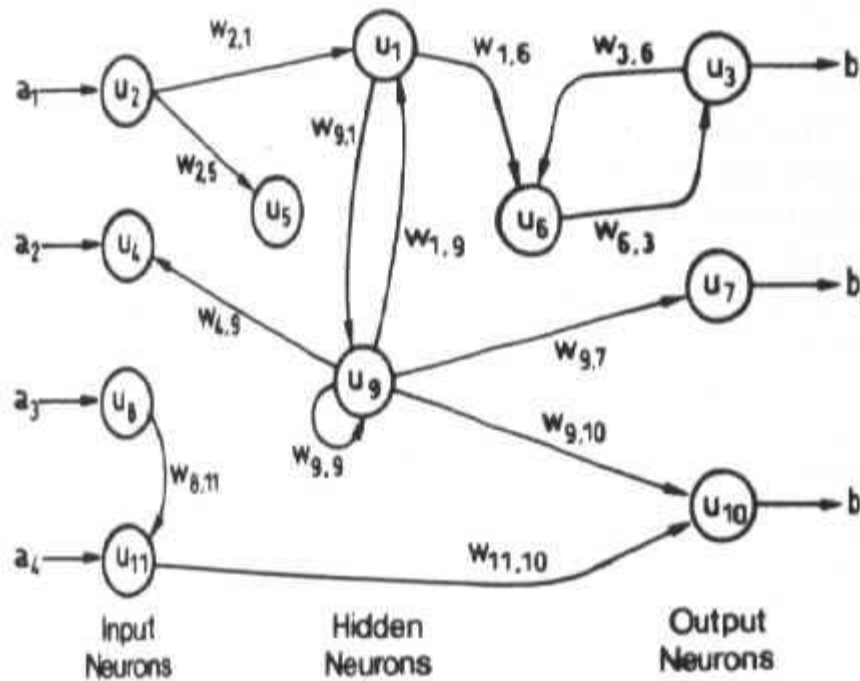


Figure 2.6: A complicated network

2.4 Learning Process paradigms

Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights. [9]

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

- 1- associative mapping** in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:

auto-association: an input pattern is associated with itself and the states of input and output units coincide.

This is used to provide pattern completion, i.e. to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.

Hetero-association: is related to two recall mechanisms:

- *nearest-neighbor* recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.
- *interpolative* recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, i.e. when there is a fixed set of categories into which the input patterns are to be classified.

2- regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation. [9]

2.5 Machine Learning Algorithms

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data

mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:

1- Supervised learning: which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning, an important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error.

As special cases, the input signal can be only partially available, or restricted to special feedback:

- Semi-supervised learning: the computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.
- Active learning: the computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.
- Reinforcement learning: training data (in form of rewards and punishments) is given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

2- Unsupervised learning: uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning.

Simply in unsupervised learning no labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). [10]

2.6 Transfer Functions (Activation Functions)

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. Most common functions are:

- linear (or ramp)
- threshold
- sigmoid

For **linear units**, the output activity is proportional to the total weighted output.

For **threshold units**, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For **sigmoid units**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations. [9]

To make a neural network that performs some specific task, we must choose how the units are connected to one another and we must set the weights on the

connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. We determine how closely the actual output of the network matches the desired output.
3. We change the weight of each connection so that the network produces a better approximation of the desired output. [9]

2.7 Applications of Neural Networks

Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries, since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing

ANN are also could be used in the following specific paradigms: recognition of speakers in communications, diagnosis of hepatitis, undersea mine detection, recovery of telecommunications from faulty software, interpretation of multi-meaning Chinese words, texture analysis, three-dimensional object recognition; hand-written word recognition, and facial recognition. [11]

2.7.1 Neural networks in Medicine

Artificial Neural Networks (ANN) are currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modelling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans also known as computed tomography scan, ultrasonic scans, etc..).

Neural Networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease is not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quality'. The examples need to be selected very carefully if the system is to perform reliably and efficiently. [11]

2.7.2 Neural Networks in Business

Business is a diversified field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis.

There is some potential for using neural networks for business purposes, including resource allocation and scheduling. There is also a strong potential for using neural networks for database mining, that is, searching for patterns implicit within the explicitly stored information in databases.

Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling. [11]

2.7.3 Neural Networks in Marketing

There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feedforward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution. The system is used to monitor and recommend booking advice for each departure. Such information has a direct impact on the profitability of an airline and can provide a technological advantage for users of the system.

While it is significant that neural networks have been applied to this problem, it is also important to see that this intelligent technology can be integrated with expert systems and other approaches to make a functional system. Neural networks were used to discover the influence of undefined interactions by the various variables. While these interactions were not defined, they were used by the neural system to develop useful conclusions. It is also noteworthy to see that neural networks can influence the bottom line. [11]

2.7.4 Neural Networks in Control Systems

Today, there is a constant need to provide better control of more complex (probably nonlinear) systems, over a wide range of uncertainty. Artificial neural networks offer a large number of attractive features for the area of control systems:

- The ability to perform arbitrary nonlinear mappings makes them a cost-efficient tool to synthesize accurate forward and inverse models of nonlinear dynamical systems, allowing traditional control schemes to be extended to the control of nonlinear plants. This can be done without the need for detailed knowledge of the plant. [12]

- The ability to create arbitrary decision regions means that they have the potential to be applied to fault detection problems. Exploiting this property, a possible use of ANNs is as control managers, deciding which control algorithm to employ based on current operational conditions.
- As their training can be affected on-line or off-line, the applications considered in the last two points can be designed off-line and afterwards used in an adaptive scheme, if so desired.
- Neural networks are massive parallel computation structures. This allows calculations to be performed at a high speed, making real-time implementations feasible. Development of fast architectures further reduce computation time.
- Neural networks can also provide, significant fault tolerance, since damage to a few weights need not significantly impair the overall performance. [12]

During the last years, a large number of applications of ANN to robotics, failure detection systems, nonlinear systems identification and control of nonlinear dynamical systems have been proposed.

Robots are nonlinear, complicated structures. It is therefore not surprising that robotics was one of the first fields where ANNs were applied. It is still one of the most active areas for applications of artificial neural networks. Applying Multilayer perceptrons to the kinematic control of a robot arm. Using Hopfield networks to solve the inverse kinematics problem. Implementing a neural controller which learns hand-eye coordination from its own experience. Employing Multilayer perceptrons with time delay elements in the hidden layer for impact control of robotic manipulators.

Artificial neural networks have also been applied to sensor failure detection and diagnosis. In these applications, the ability of neural networks for creating arbitrary decision regions is exploited. As examples employing MLPs for the sensor failure detection in process control systems, obtaining promising results, when compared to traditional methods. Using of neural networks, in the three different capacities

of identifiers, pattern recognizers and controllers, to detect, classify and recover from faults in control systems. [12]

2.8 Advantages of Neural Networks

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage. [8]

2.9 Neural Networks versus Conventional Computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But

computers would be so much more useful if they could do things that we don't exactly know how to do. [9]

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements(neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency. [9]

To better understand artificial neural computing, it is important to know first how a conventional 'serial' computer and its software process information. A serial computer has a central processor that can address an array of memory locations where data and instructions are stored. Computations are made by the processor reading an instruction as well as any data the instruction requires from memory addresses, the instruction is then executed and the results are saved in a specified

memory location as required. In a serial system (and a standard parallel one as well) the computational steps are deterministic, sequential and logical, and the state of a given variable can be tracked from one operation to another. [9]

In comparison, ANNs are not sequential or necessarily deterministic. There are no complex central processors, rather there are many simple ones which generally do nothing more than take the weighted sum of their inputs from other processors. ANNs do not execute programmed instructions; they respond in parallel (either simulated or actual) to the pattern of inputs presented to it. There are also no separate memory addresses for storing data. Instead, information is contained in the overall activation 'state' of the network. 'Knowledge' is thus represented by the network itself, which is quite literally more than the sum of its individual components.

So, depending on the nature of the application and the strength of the internal data patterns you can generally expect a network to train quite well. This applies to problems where the relationships may be quite dynamic or non-linear. ANNs provide an analytical alternative to conventional techniques which are often limited by strict assumptions of normality, linearity, variable independence etc.

Because an ANN can capture many kinds of relationships it allows the user to quickly and relatively easily model phenomena which otherwise may have been very difficult or impossible to explain otherwise. [9]

Chapter 3

Backpropagation

3.1 Introduction

Backpropagation is a method used in Artificial Neural Networks using gradient descent to calculate a gradient that is needed in the calculation of the weights to be used in the network. It is commonly used to train deep neural networks, a term referring to neural networks with more than one hidden layer.

The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning, making it possible to use neural networks to solve problems which had previously been insolvable.

It is a special case of an older and more general technique called automatic differentiation. It is also called “backward propagation of errors”; because the error is calculated at the output and distributed back through the network layers, this backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer, the motivation for backpropagation is to train a multi-layered neural net so it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output. [13]

It is considered to be a supervised learning method; the goal of any supervised learning algorithm is to find a function that best maps a set of inputs to their correct output. An example would be a classification task, where the input is an image of an animal, and the correct output is the type of animal (e.g. Dog, cat, giraffe, lion, zebra, etc.).

Backpropagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and speech recognition. It is considered an efficient algorithm, and modern implementations take advantage of specialized GPUs to further improve performance. [14]

3.2 Gradient Descent

The gradient of the function f with respect to the variable x defines how much the value of f will change with a unit increase/decrease in the value of x . Generally speaking, a function may be dependent on a variety of parameters, so you take a partial derivative when computing the gradient.

Consider you are on a hill like the one shown in (Figure 3.1) below:

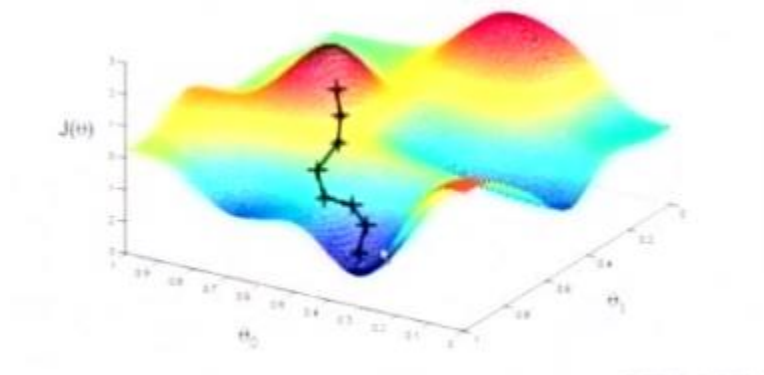


Figure 3.1: Gradient Descent example

The red regions correspond to places of higher function value, while the blue regions correspond to low values, (θ_0, θ_1) represents the dimensions, and $J(\theta)$ is the target function. You want to reach the lowest point possible. If you are standing on the red hill, and you can only see a small part of the terrain around you, what will you do? You will naturally take steps in the direction in which the terrain slopes down. With each step you take, you will scan your immediate neighborhood (which is visible to you), and go in the direction that shows the steepest descent.

In mathematical terms, this ‘direction’ is nothing but the gradient of your height (the target function you are trying to minimize), with respect to your location (x, y) (which are the parameters you are fine tuning).

The overall direction in which you move, will be dependent on the gradients with respect to both dimensions, in example $\frac{\partial height}{\partial x}$ and $\frac{\partial height}{\partial y}$. That’s pretty much what gradient descent does.

Its called so, because you are *descending* from a higher value of the target function to a lower value, by following the direction that corresponds to the lowest *gradient* (and consequently the steepest decrease in function value).

One thing to note is that you would always want to move in the direction shown by the *negation* of the gradient. Therefore, to reduce $f(x)$, you would decrease the value of x . The opposite is true in scenarios where you want to maximize the target function. In those cases, you move in the direction of the gradient itself. [15]

3.3 Backpropagation Requirements

1) **Dataset:** a neural network is initially trained, or fed large amounts of data, in the same way that we learn from experience in our lives, so neural networks require data to learn. In most cases, the more data that can be thrown at a neural network, the more accurate it will become.

2) **Feedforward neural network:** which information travels in only one direction from input to output. since you are going from the input layer to the output layer (through the hidden layers if any). These types of models are simply called feed-forward linear layers, and for the sake of this Project, we will use the sigmoid function $g(x)$ as an activation function.

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

3) **Error function (loss function):** this function defines how much the output of our model differs from the required output. (In this Project, we will use eq (2) the Mean-Squared Error (MSE) function).

$$E = \frac{1}{2}(t_0 - a_0)^2 \quad (2)$$

the expected output (*desired*) is t_0 . The *actual* output is a_0 .

3.4 Training Steps

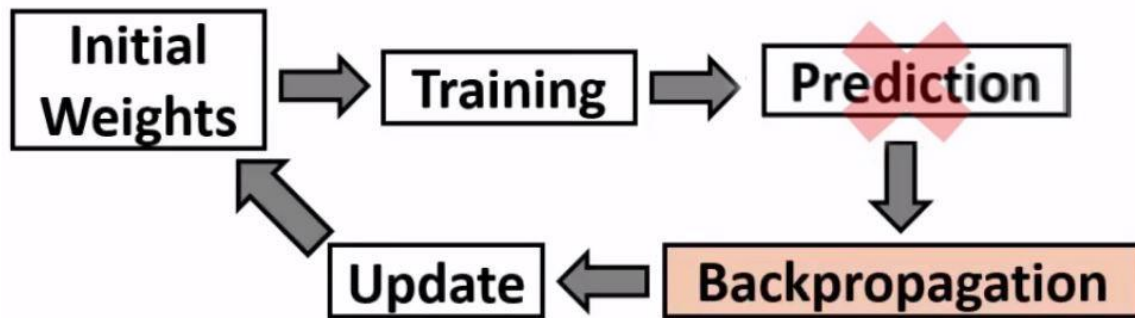


Figure 3.2: Training diagram

Step 1- Model initialization

The first step of the learning, is to start from somewhere, the initial hypothesis. Like in genetic algorithms and evolution theory, neural networks can start from anywhere. Thus, a random initialization of the model is a common practice. The rationale behind is that from wherever we start, if we are perseverant enough and through an iterative learning process, we can reach the pseudo-ideal model.

In order to give an analogy, take for instance a person who has never played football in his life. The very first time he tries to shoot the ball, he can just shoot it randomly.

we are exploring which model of the generic form ($y = \omega \cdot x$) can fit the best the current dataset. Where ω is called the *weights* of the network and can be initialized randomly. [16]

Step 2- Forward propagate

The natural step to do after initializing the model at random, is to check its performance.

We start from the input we have, we pass them through the network layer and calculate the actual output of the model straightforwardly.

This step is called forward-propagation, because the calculation flow is going in the natural forward direction from the input -> through the neural network -> to the output.

If you have multiple output neurons, the error values corresponding to each will affect the weights appropriately. [16]

Step 3- Loss function

At this stage, in one hand, we have the actual output of the randomly initialized neural network a_0 . on the other hand, we have the desired output we would like the network to learn t_0 .

In the first time, the actual output will be the final position of the ball, the desired output would be that the ball goes inside the goal. In the beginning, our player is just shooting randomly. Let's say the ball went most of the time, to the right side of the goal. What he can learn from this, is that he needs to shoot a bit more to the left next time he trains.

In order to be able to generalize to any problem, we define what we call: loss function. Basically, it is a performance metric on how well the neural network

manages to reach its goal of generating outputs as close as possible to the desired values.

In statistics, the mean squared error (MSE) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors, the average squared difference between the estimated values and what is estimated.

The MSE assesses the quality of an estimator (i.e. a mathematical function mapping a sample of data to an estimate of a parameter of the population from which the data is sampled) or a predictor (i.e., a function mapping arbitrary inputs to a sample of values of some random variable). The definition of an MSE differs according to whether one is describing an estimator or a predictor. [16]

If $\hat{Y}$ is a vector of n predictions generated from a sample of n data points on all variables, and Y is the vector of observed values of the variable being predicted, then the within-sample MSE of the predictor is computed as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (3)$$

Applying that on our situation we got eq (2):

$$E = \frac{1}{2} (t_0 - a_0)^2$$

Squared error loss is one of the most widely used loss functions in statistics, though its widespread use stems more from mathematical convenience than considerations of actual loss in applications. The mathematical benefits of mean squared error are particularly evident in its use at analyzing the performance of linear regression, as it allows one to partition the variation in a dataset into variation explained by the model and variation explained by randomness.

As a summary, the loss function is an error metric, that gives an indicator on how much precision we lose, if we replace the real desired output by the actual output generated by our trained neural network model. That's why it's called loss!

Simply speaking, the machine learning goal becomes then to minimize the loss function (to reach as close as possible to 0).

We can just transform our machine learning problem now to an optimization process that aims to minimize this loss function. [17]

Step 4- Differentiation

Obviously, we can use any optimization technique that modifies the internal weights of neural networks in order to minimize the total loss function that we previously defined. These techniques can include genetic algorithms or greedy search or even a simple brute-force search.

Luckily for us, there is a powerful concept in mathematics that can guide us how to optimize the weights called differentiation. Basically, it deals with the derivative of the loss function. In mathematics, the derivative of a function at a certain point, gives the rate or the speed of which this function is changing its values at this point.

what we really care about is the rate of which the error changes relatively to the changes on the weight, so the advantage of using the mathematical derivative is that it is much faster and more precise to calculate (less floating-point precision problems), If we initialize randomly the network, we are putting any random point on this curve. The learning process algorithm would be like this:

- check the derivative.

- If it is positive, meaning the error increases if we increase the weights, then we should decrease the weight.
- If it's negative, meaning the error decreases if we increase the weights, then we should increase the weight.
- If it's 0, we do nothing, we reach our stable point.

In a simple matter as shown in (Figure 3.3) while w is the weight and $J(w)$ is the energy level, we are designing a process that acts like gravity. No matter where we randomly initialize the ball on this error function curve, there is a kind of force field that drives the ball back to the lowest energy level of ground 0. [16]

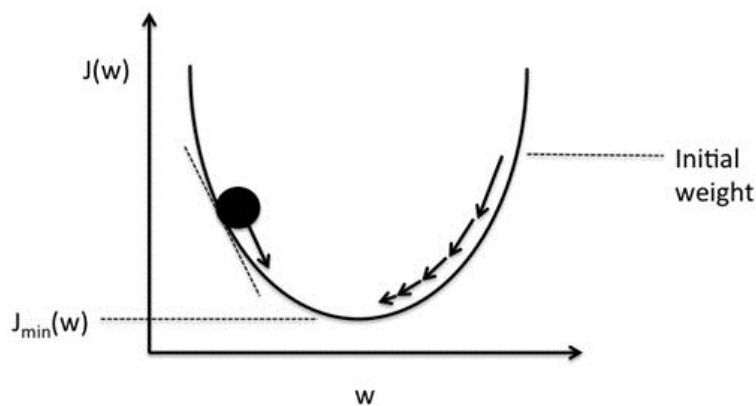


Figure 3.3: Optimization operation.

Step 5- Back-propagation

We have the starting point of errors, which is the loss function, and we know how to derivate it, and if we know how to derivate each function from the composition, we can propagate back the error from the end to the start, (Figure 3.4) show us the diagram of forward network then the diagram of backpropagation method.

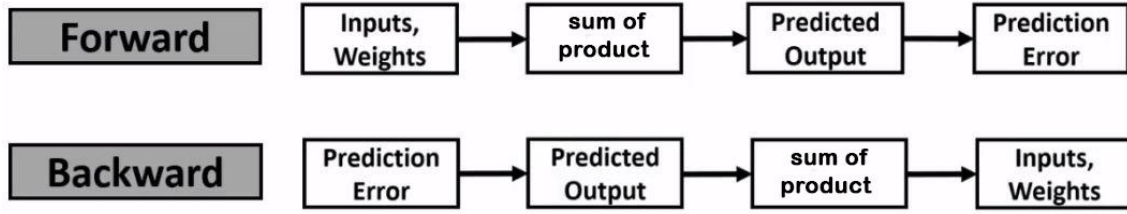


Figure 3.4: Weights calculating schema using Backpropagation

If we create a library of differentiable functions or layers where for each function we know how to forward-propagate (by directly applying the function) and how to back-propagate (by knowing the derivative of the function), we can compose any complex neural network. We only need to keep a stack of the function calls during the forward pass and their parameters, in order to know the way back to backpropagate the errors using the derivatives of these functions. This can be done by de-stacking through the function calls. This technique is called auto-differentiation, and requires only that each function is provided with the implementation of its derivative. In a future blog post, we will explain how to accelerate auto-differentiation by implementing basic mathematical operations over matrices.

Now any layer can forward its results to many other layers, in this case, in order to do backpropagation, we sum the errors coming from all the target layers. Thus, our calculation stack can become a complex calculation graph. [16]

backpropagation reduces the error function E by taking help of the gradients with respect to individual weights $\frac{\partial E}{\partial \omega_{i,j}}$.

So, as a first step, let's try to understand how we will calculate the gradient $\frac{\partial E}{\partial \omega_{i,j}}$ for a given training sample. Remember that $\omega_{i,j}$ is the weightage given. Using the chain rule in Leibniz's form as explained in (Figure 3.5) we get eq (4) :

$$\frac{\partial E}{\partial \omega_{i,j}} = \frac{\partial E}{\partial in} \frac{\partial in}{\partial \omega_{i,j}} \quad (4)$$

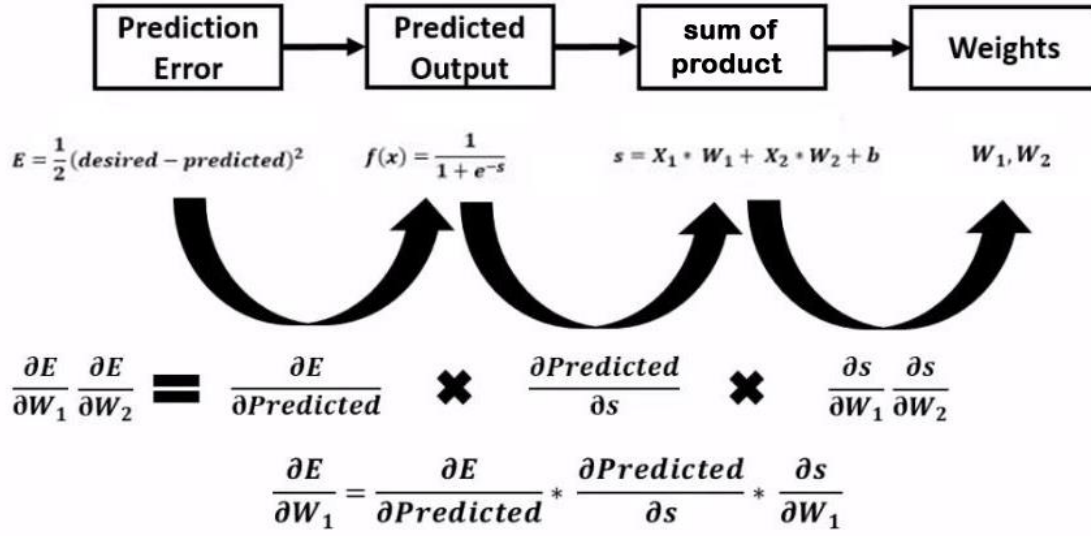


Figure 3.5: Using of chain rule

Let's say that the weightage assigned by neuron N_i to N_j is $\omega_{i,j}$. As the first step,

N_i computes a weighted sum of all its inputs. Let's call it in_i and a_j is the activation function in the neuron N_j . Therefore:

$$in_i = \sum_{k \in \text{Inputs}(N_i)} \omega_{i,j} a_j \quad (5)$$

From Equations (4) and (5), and using basic differentiation:

$$\frac{\partial E}{\partial \omega_{i,j}} = a_j \frac{\partial E}{\partial in_i} \quad (6)$$

From the basic definition of the activation function (and the appropriate current weight values), we already know the value of a_j . Now consider the second term in the above product. Using the chain rule again on it, we get eq (7):

$$\frac{\partial E}{\partial in_i} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial in_i} \quad (7)$$

This is where the whole advantage of using the sigmoid activation function comes into play. The derivative of a sigmoid function with respect to its input, corresponds to a nice expression, you can easily verify that:

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

$$\frac{dg}{dx} = g(x)(1 - g(x)) \quad (8)$$

From Equations 7 and 8 we get eq (9):

$$\frac{\partial E}{\partial in_i} = a_i(1 - a_i) \frac{\partial E}{\partial a_i} \quad (9)$$

Putting the information in Equations 6 and 9 together,

$$\frac{\partial E}{\partial \omega_{i,j}} = a_j a_i(1 - a_i) \frac{\partial E}{\partial a_i} \quad (10)$$

Look at the last term. There are two possible cases for neuron N_i (N_i cannot be an input neuron, since it is receiving input from another neuron):

I. N_i is an **output neuron**. In this case, the second term on the right-hand side is pretty simple, from Eq (2):

$$E = \frac{1}{2}(t_0 - a_0)^2$$

$$\frac{\partial E}{\partial a_i} = -(t_i - a_i) \quad (11)$$

where t_0 is the expected output from N_i (for the current training sample).

2. N_i is an **inner neuron**. Consider all neurons N_k , for whom N_i acts as an input. Since we are propagating backward, you can therefore assume that we know the values $\frac{\partial E}{\partial a_k}$ for all of them. In fact, since the output of N_i affects the outputs of all these N_k s, the expression for $\frac{\partial E}{\partial a_i}$ will involve all of them.

Using simple chain rule,

$$\frac{\partial E}{\partial a_i} = \sum_k \frac{\partial E}{\partial in_k} \frac{\partial in_k}{\partial a_i} \quad (12)$$

From Equations 5 and 9

$$\frac{\partial E}{\partial a_i} = \sum_k \omega_{k,i} a_k (1 - a_k) \frac{\partial E}{\partial a_k} \quad (13)$$

Equations (10) with (11) & (13) are pretty much define the gradient values $\frac{\partial E}{\partial \omega_{i,j}}$ for all weights in ANN. [16]

Step 6- Weight update

This updating occurs first for the out-most weights, and then for lower layers, the intuition behind this, is the fact that the errors with respect to the outmost neurons are known first, and you derive the errors of the inner neurons from them, each layer's errors are calculated from the errors of the neurons they provide input to.

Since there are lot of non-linearities, any big change in weights will lead to a chaotic behavior.

As we presented earlier, the derivative is just the rate of which the error changes relatively to the weight changes, we should not forget that the derivative is only local at the point where we are calculating the derivative.

Thus, as a general rule of weight updates is the delta rule:

*New weight = old weight - Derivative Rate * learning rate*

$$W_{inew} = W_{iold} - \eta * \frac{\partial E}{\partial W_i} \quad (14)$$

- W_{inew} : new updated weights.
- W_{iold} : current weights.
- η : learning rate.
- $\frac{\partial E}{\partial W_i}$: derivative rate.

The learning rate is introduced as a constant (usually small), in order to force the weight to get updated very smoothly and slowly (to avoid big steps and chaotic behavior).

In order to validate this equation:

- If the derivative rate is positive, it means that an increase in weight will increase the error, thus the new weight should be smaller.
- If the derivative rate is negative, it means that an increase in weight will decrease the error, thus we need to increase the weights.
- If the derivative is 0, it means that we are in a stable minimum. Thus, no update on the weights is needed -> we reached a stable state.

Now several weight update methods exist. These methods are often called “optimizers”.

Any optimizer can work with these 3 modes (full online / full-batch / mini-batch).

1. Online Mode

In this mode, the weights are updated after every training sample. In other words, the forward propagation as well as the backward propagation phases are run for each training sample, every single time. This mode is appropriate if you are

receiving your training data sequentially. However, if your training set is large, this mode of training can be pretty time-consuming. Moreover, it is not appropriate to update weights after every sample (in most cases), due to the possible presence of outliers.

2. Batch Mode

In this mode, the weights are updated after accumulating gradients with respect to all samples in the training set. In other words, the forward propagation is run with respect to all samples, and the backward phase is run just once with respect to the accumulated results of the entire batch. This is done for multiple iterations (obviously). The batch mode of training is thus more resistant to variance in the training dataset.

3. Stochastic Mode “mini-batch”

The stochastic mode adds randomization to the mix. In the mini-batch method of stochastic learning, small sets of samples are chosen randomly from the dataset, and training is done in batch mode over these mini-batches. Stochastic mode with mini-batches provides a nice trade-off between the pros and cons of online and batch modes of training. Moreover, since you are randomly choosing the mini batches (instead of organizing your dataset into pre-defined subsets), you avoid getting stuck in local optima. This is mainly because you are providing weight updates with respect to a potentially different subset of training samples during every iteration. [17]

Step 7- Iterate until convergence

This is very similar to genetic algorithms where after each generation we apply a small mutation rate and the fittest survives. In neural network, after each iteration, the gradient descent force updates the weights towards less and less global loss function.

Number of iterations are needed to converge depends on:

- how strong the learning rate we are applying, High learning rate means faster learning, but with higher chance of instability.
- the meta-parameters of the network (how many layers, how complex the non-linear functions are). The more it has variables the more it takes time to converge, but the higher precision it can reach.
- the optimization method uses, some weight updates rule is proven to be faster than others.
- the random initialization of the network. If the target was “ $W=2$ ” Maybe with some luck you will initialize the network with “ $W=1.99$ ” then you are only one step away from the optimal solution.
- the quality of the training set. If the input and output have no correlation between each other, the neural network will not do magic and can't learn a random correlation. [17]

Chapter 4

Digital Image Processing

4.1 Introduction

The term digital image processing generally refers to processing of a two-dimensional picture by a digital computer.

In a broader context, it implies digital processing of any two-dimensional data or extract information from the image digitally by the use of computer algorithm. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and the preservation of original data precision.

Digital image processing is always an interesting field as it gives improved pictorial information for human interpretation and processing of image data for storage, transmission, and representation for machine perception. Image Processing is a technique to enhance raw images received from cameras/sensors placed on satellites, space probes and aircrafts or pictures taken in normal day-to-day life for various applications. This field of image processing significantly improved in recent times and extended to various fields of science and technology. The image processing mainly deals with image acquisition, Image enhancement, image segmentation, feature extraction, image classification etc.

Image processing or digital image manipulation is one of the greatest advantages of digital radiography (DR). Preprocessing depends on the modality and corrects for system irregularities such as differential light detection efficiency, dead pixels, or dark noise. Processing is manipulation of the raw data just after acquisition. It is generally proprietary and specific to the DR vendor but encompasses manipulations such as unsharp mask filtering within two or more spatial frequency bands, histogram sliding and stretching, and gray scale rendition or lookup table application. These processing steps have a profound effect on the final appearance of the radiograph, but they can also lead to artifacts unique to digital systems. Postprocessing refers to manipulation of the final appearance of the radiograph by the end-user and does not involve alteration of the raw data. [18]

The noise or irregularity may creep into the image either during its formation or during transformation, for mathematical analysis, an image may be defined as a two-dimensional function $f(x, y)$ where x and y are spatial (plane) coordinates, and the amplitude of function at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x , y , and the intensity values of f are all finite, discrete quantities, we call the image a digital image. It is very important that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are called picture elements, image elements, and pixels. Pixel is the most widely used term to denote the elements of a digital image.

Various techniques have been developed in Image Processing during the last four to five decades. Most of the techniques are developed for enhancing images obtained from unmanned spacecrafts, space probes and military reconnaissance flights. Image Processing systems are becoming popular due to easy availability of powerful personnel computers, large size memory devices, graphics software etc. [19]

4.2 Digital Image Processing Algorithms Usages

- Convert signals from an image sensor into digital images
- Improve clarity, and remove noise and other artifacts
- Prepare images for display or printing
- Compress images for communication across a network
- Extract the size, scale, or number of objects in a scene, like counting circular objects as shown in (Figure 4.1).

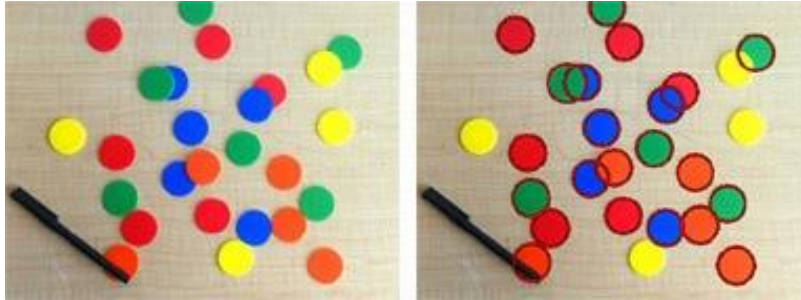


Figure 4.1: Counting circular objects in an image.

4.3 Image Processing Techniques

The techniques of image processing include:

- Image preprocessing
- Image segmentation
- Feature extraction
- Image classification

4.3.1 Image Preprocessing

In image preprocessing, image data recorded by sensors on a satellite restrain errors related to geometry and brightness values of the pixels. These errors are corrected using appropriate mathematical models which are either definite or statistical models. Image enhancement is the modification of image by changing the pixel brightness values to improve its visual impact.

Image enhancement involves a collection of techniques that are used to improve the visual appearance of an image, or to convert the image to a form which is better suited for human or machine interpretation.

Sometimes images obtained from satellites and conventional and digital cameras lack in contrast and brightness because of the limitations of imaging sub systems and illumination conditions while capturing image. Images may have different types of noise.

In image enhancement, the goal is to accentuate certain image features for subsequent analysis or for image display.

Examples include contrast and edge enhancement, pseudo-coloring, noise filtering, sharpening, and magnifying. Image enhancement is useful in feature extraction, image analysis and an image display. The enhancement process itself does not increase the inherent information content in the data. It simply emphasizes certain specified image characteristics. Enhancement algorithms are generally interactive and application dependent. [20]

Some of the enhancement techniques are:

a. Contrast Stretching

Some images (e.g. over water bodies, deserts, dense forests, snow, clouds and under hazy conditions over heterogeneous regions) are homogeneous i.e., they do not have much change in their levels. In terms of histogram representation, they are characterized as the occurrence of very narrow peaks. The homogeneity can also be due to the incorrect illumination of the scene. Ultimately the images hence obtained are not easily interpretable due to poor human perceptibility. This is because there exists only a narrow range of gray-levels in the image having provision for wider range of gray-levels. The contrast stretching methods are designed exclusively for frequently encountered situations. Different stretching techniques have been developed to stretch the narrow range to the whole of the available dynamic range, (Figure 4.2) clarify the effect of contrast stretching over a satellite image. [21]

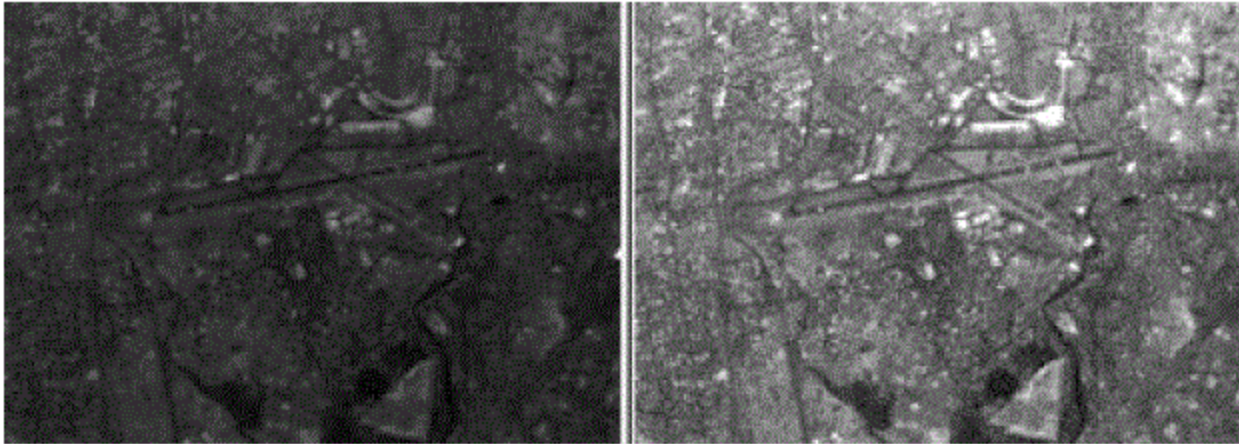


Figure 4.2: contrast stretching

b. Noise Filtering

Noise Filtering is used to filter the unnecessary information from an image. It is also used to remove various types of noises from the images. Mostly this feature is interactive. Various filters like low pass, high pass, mean, median etc., are available.

Filtering also helps to blur or sharpen the image such as shown in (Figure 4.3). Digital image processing is depending upon the computer vision. There are two types of operation in dip. Low and middle level operation.

Smoothing frequency which is helps to smoothen the image with some filters such as Ideal low pass filters, Butterworth Low pass filter, Gaussian low pass Filters. following the same, we can have filters to sharpen the images such as, Ideal High pass filters, Butterworth high pass filters, Gaussian High pass filters, High-boost filtering and High-frequency Emphasis filtering. [22]

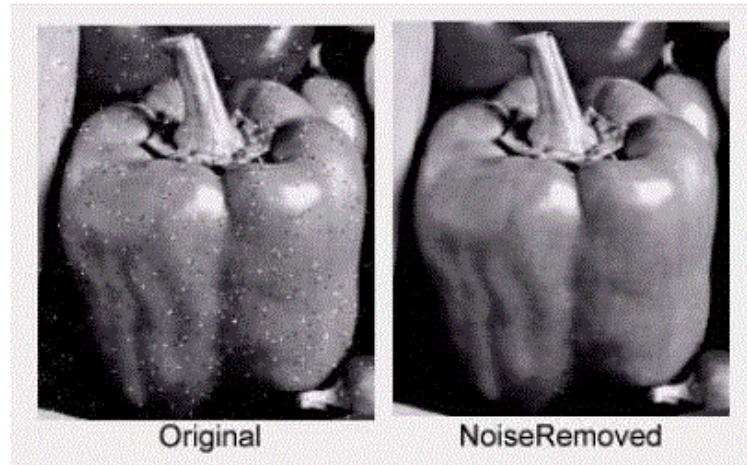


Figure 4.3: Noise Removal

c. Histogram modification

Histogram Modification Histogram has a lot of importance in image enhancement. It reflects the characteristics of image. By modifying the histogram, image characteristics can be modified. One such example is Histogram Equalization. Histogram equalization is a nonlinear stretch that redistributes pixel values so that there is approximately the same number of pixels with each value within a range. The result approximates a flat histogram. Therefore, contrast is increased at the peaks and lessened at the tails. [22]

4.3.2 Image Segmentation

Segmentation is one of the key problems in image processing. Image segmentation is the process that subdivides an image into its constituent parts or objects. The level to which this subdivision is carried out depends on the problem being solved, i.e., the segmentation should stop when the objects of interest in an application have been isolated e.g., in autonomous air-to-ground target acquisition, suppose our interest lies in identifying vehicles on a road, the first step is to segment the road from the image and then to segment the contents of the road down to potential vehicles. Image thresholding techniques are used for image segmentation.

After thresholding a binary image is formed where all object pixels have one gray level and all background pixels have another - generally the object pixels are 'black' and the background is 'white'. The best threshold is the one that selects all the object pixels and maps them to 'black'. Various approaches for the automatic selection of the threshold have been proposed. Thresholding can be defined as mapping of the gray scale into the binary set $\{0, 1\}$:

$$s(x, y) = \begin{cases} 0, & \text{if } g(x, y) < T(x, y) \\ 1, & \text{if } g(x, y) \geq T(x, y) \end{cases}$$

where $S(x, y)$ is the value of the segmented image, $g(x, y)$ is the gray level of the pixel (x, y) and $T(x, y)$ is the threshold value at the coordinates (x, y) . In the simplest case $T(x, y)$ is coordinate independent and a constant for the whole image. It can be selected, for instance, on the basis of the gray level histogram. When the histogram has two pronounced maxima, which reflect gray levels of object(s) and background, it is possible to select a single threshold for the entire image. A method which is based on this idea and uses a correlation criterion to select the best threshold, is described below. Sometimes gray level histograms have only one maximum. This can be caused, e.g., by inhomogeneous illumination of various regions of the image. In such case it is impossible to select a single thresholding value for the entire image and a local binarization technique must be applied. General methods to solve the problem of binarization of inhomogeneously illuminated images, however, are not available. [23]

Segmentation of images involves sometimes not only the discrimination between objects and the background, but also separation between different regions. One method for such separation is known as watershed segmentation. [23]

4.3.3 Feature Extraction

The feature extraction techniques are developed to extract features in synthetic aperture radar images. This technique extracts high-level features needed in order to perform classification of targets. Features are those items which uniquely describe a target, such as size, shape, composition, location etc. Segmentation techniques are used to isolate the desired object from the scene so that measurements can be made on it subsequently. Quantitative measurements of object features allow classification and description of the image. When the pre-processing and the desired level of segmentation has been achieved, some feature extraction technique is applied to the segments to obtain features, which is followed by application of classification and post processing techniques. It is essential to focus on the feature extraction phase as it has an observable impact on the efficiency of the recognition system. Feature selection of a feature extraction method is the single most important factor in achieving high recognition performance. Feature extraction has been given as “extracting from the raw data information that is most suitable for classification purposes, while minimizing the within class pattern variability and enhancing the between class pattern variability”. Thus, selection of a suitable feature extraction technique according to the input to be applied needs to be done with utmost care. Taking into consideration all these factors, it becomes essential to look at the various available techniques for feature extraction in a given domain, covering vast possibilities of cases. [24]

4.3.4 Image Classification

The intent of the classification process is to categorize all pixels in a digital image into one of several land cover classes, or "themes". This categorized data may then be used to produce thematic maps of the land cover present in an image. Normally, multispectral data are used to perform the classification and, indeed, the spectral pattern present within the data for each pixel is used as the numerical basis for categorization (Lillesand and Kiefer, 1994). The objective of image classification is to identify and portray, as a unique gray level (or color), the features occurring in an image in terms of the object or type of land cover these features actually represent on the ground. Simply Image classification is the labeling of a pixel or a group of pixels based on its grey value. [21]

Classification is one of the most often used methods of information extraction. In Classification, usually multiple features are used for a set of pixels i.e., many images of a particular object are needed. In Remote Sensing area, this procedure assumes that the imagery of a specific geographic area is collected in multiple regions of the electromagnetic spectrum and is in good registration, as noticed in (Figure 4.4). Most of the information extraction techniques rely on analysis of the spectral reflectance properties of such imagery and employ special algorithms designed to perform various types of 'spectral analysis'. The process of multispectral classification can be performed using either of the two methods: Supervised or Unsupervised. [24]

In Supervised classification, the identity and location of some of the land cover types such as urban, wetland, forest etc., are known as priori through a combination of field works and toposheets. The analyst attempts to locate specific sites in the remotely sensed data that represents homogeneous examples of these land cover types. These areas are commonly referred as 'Training sites' because the spectral characteristics of these known areas are used to 'train' the classification algorithm for eventual land cover mapping of remainder of the image. Multivariate statistical parameters are calculated for each training site. Every pixel both within and outside these training sites is then evaluated and assigned to a class of which it has the highest likelihood of being a member. [25]

In an Unsupervised classification, the identities of land cover types have to be specified as classes within a scene are not generally known as priori because ground truth is lacking or surface features within the scene are not well defined.

The computer is required to group pixel data into different spectral classes according to some statistically determined criteria. The comparison in medical area is the labeling of cells based on their shape, size, color and texture, which act as features. This method is also useful for MRI images. [25]

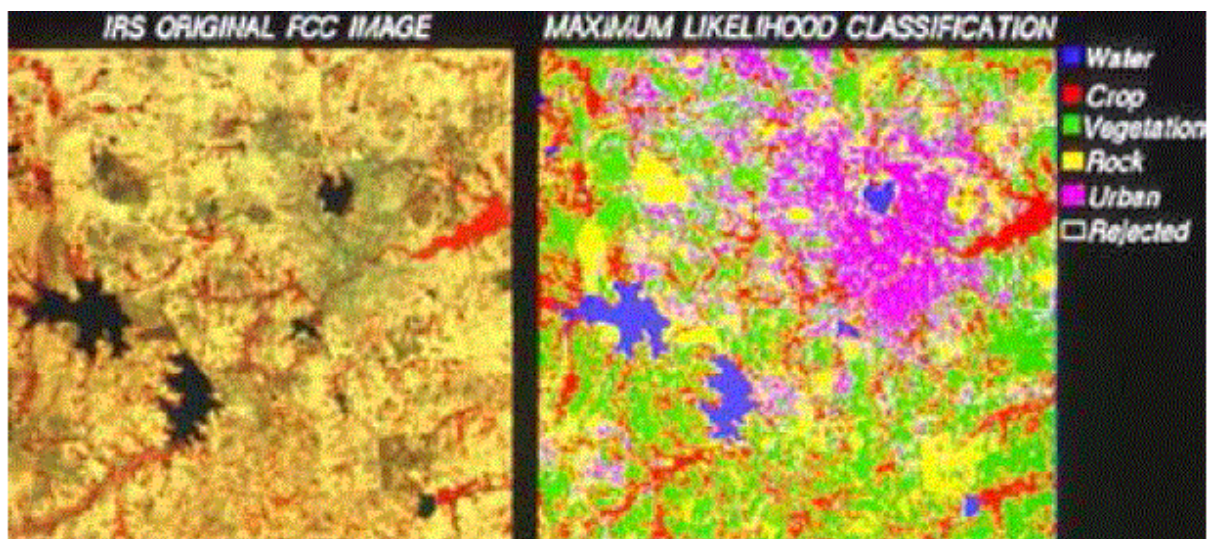


Figure 4.4: Image Classification

4.4 Image Processing Applications

Image Processing is used in various applications such as:

- Remote Sensing
- Medical Imaging
- Non-destructive Evaluation
- Forensic Studies
- Textiles

- Material Science.
- Military
- Film industry
- Document processing
- Graphic arts
- Printing Industry

Chapter 5

Software Simulation

5.1 Introduction

In this chapter a program will be coding in MATLAB command line and designing using MATLAB graphical user interface (GUI) to perform certain Digital Image processes on Images and to create a feed-forward Artificial Neural Network then train it using the Backpropagation algorithm, and finally simulate and test it over different samples.

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems. [26]

5.1.1 MATLAB GUI using GUIDE

GUIs (graphical user interfaces) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application. MATLAB® apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox™, Signal Processing Toolbox™, and Control System Toolbox™ include apps with custom user interfaces. You can also create your own custom apps, including their corresponding UIs, for others to use. [26]

GUIDE (GUI development environment) as shown in (Figure 5.1) provides tools to design user interfaces for custom apps. Using the GUIDE Layout Editor, you can graphically design your UI. GUIDE then automatically generates the MATLAB code for constructing the UI, which you can modify to program the behavior of your app. [26]

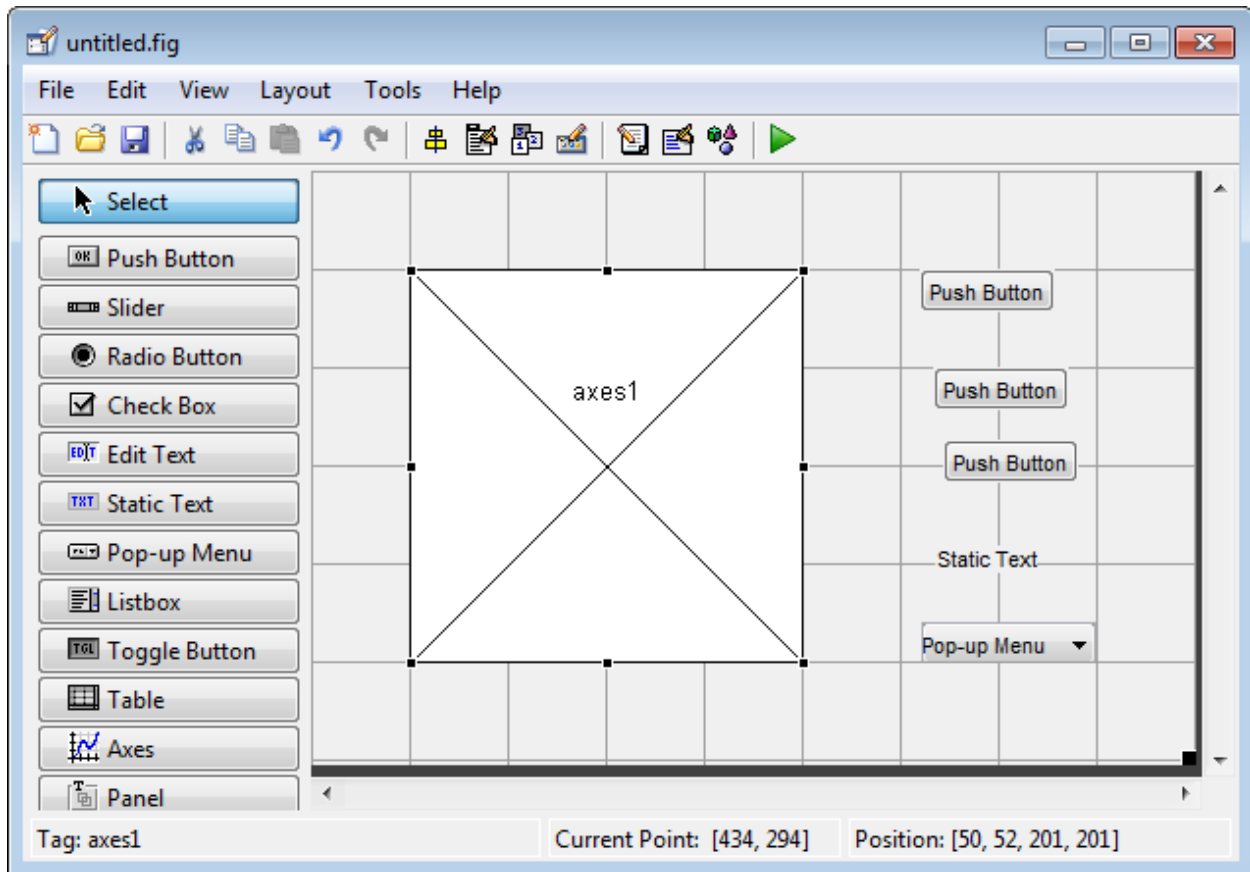


Figure 5.1: The GUIDE interface

5.2 Digital Image Processing Functions

These functions are going to be Implement on the Images that hold the data via MATLAB functions:

- **RGB2GRAY**: converts colored Red-Green-Blue (RGB) image or colormap to grayscale, by eliminating the hue and saturation information while retaining the luminance as visible in (Figure 5.2) below:



Figure 5.2: Convert RGB image to grayscale

- **IM2BW**: Converts image to binary image (black and white) from intensity, it converts this grayscale image to binary by thresholding, that would be seen in (Figure 5.3), the output binary image BW has values of 1 (white) for all pixels in the input image with luminance greater than LEVEL, and 0 (black) for all other pixels.

Calculating LEVEL done by using **GRAYTHRESH** function which calculates the Global image threshold using Otsu's method.

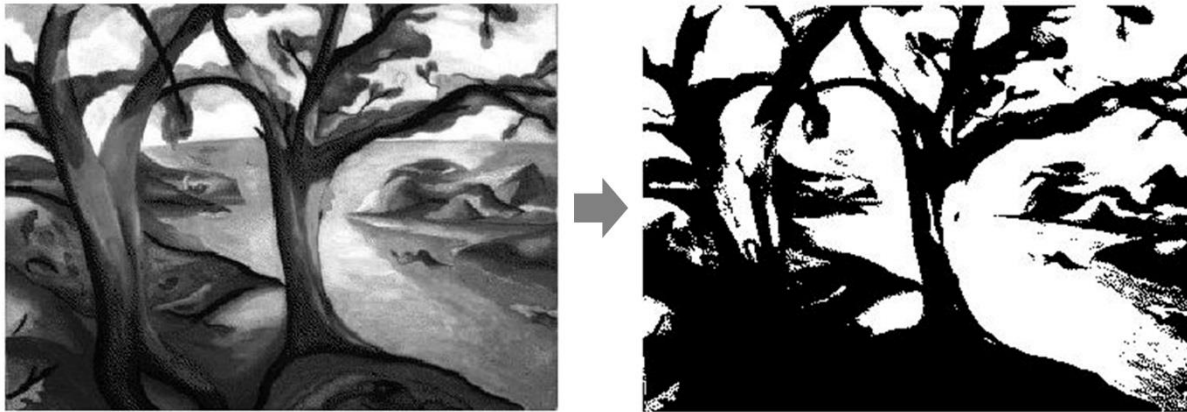


Figure 5.3: Convert grayscale image to binary image (black and white)

- **EDGE:** finds edges in intensity image.

EDGE takes an intensity or a binary image as its input, and returns a binary image BW of the same size as in (Figure 5.4), with 1's where the function finds edges in I and 0's elsewhere.

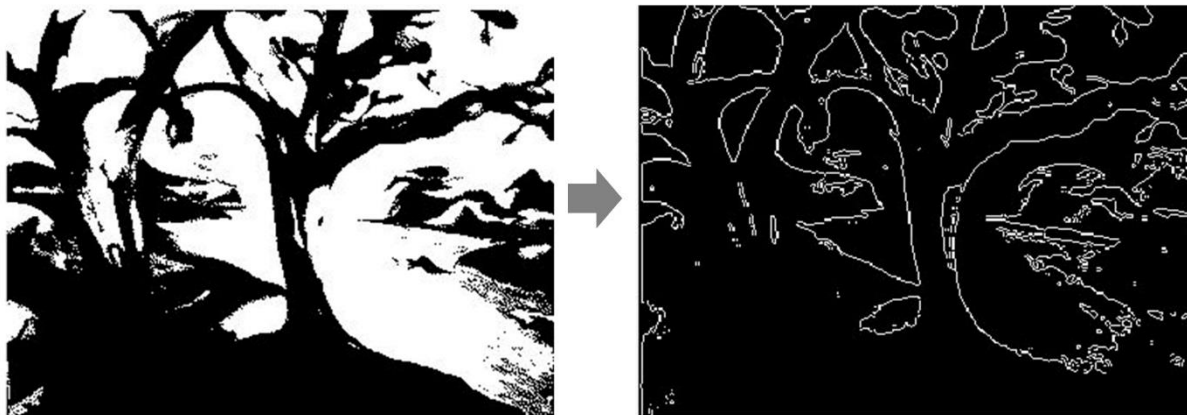


Figure 5.4: Finding the edges

- **IMDILATE**: Dilates the image.

It dilates the grayscale, binary, or packed binary image, returning the dilated image over structuring element object, such a vertical line as shown in (Figure 5.5). This effect helps us to found the boundaries of data in the image.

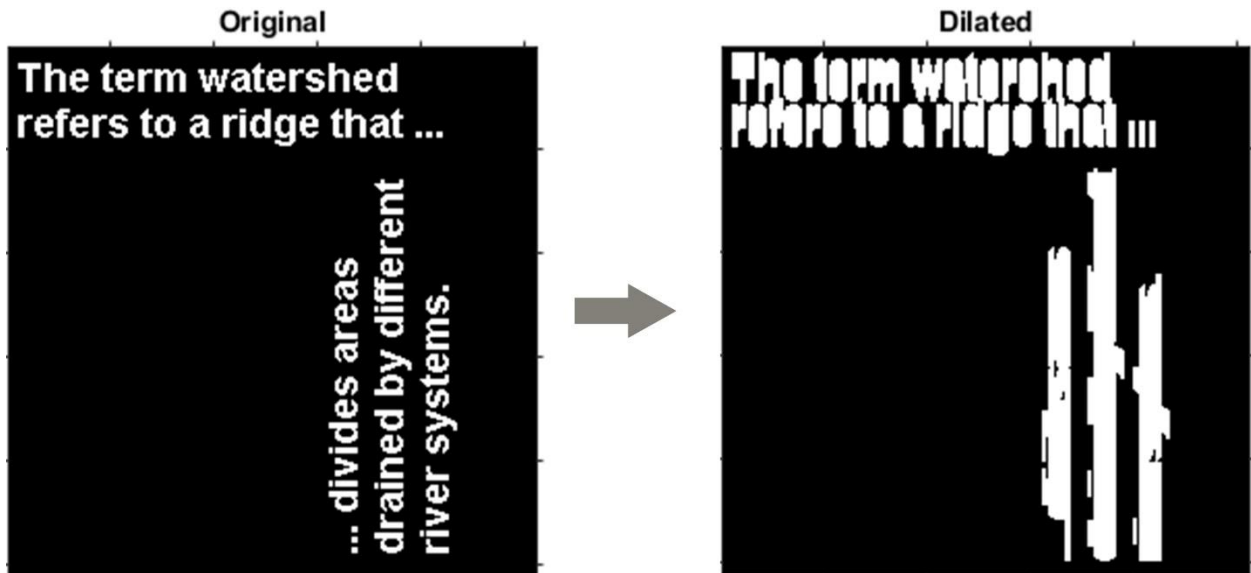


Figure 5.5: Dilating the Image

- **IMFILL**: fills the image regions and holes.

This effect removes any misunderstanding may occur by filling the space between the boundaries with the same content, as we can see in (Figure 5.6) the abnormal coin still has black color inside, this function removes any black color within any connected white shape.

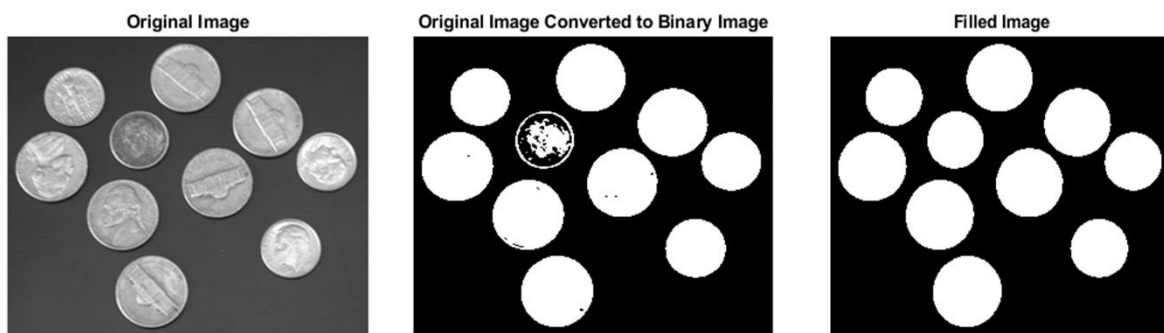


Figure 5.6: IMFILL operation

- **BWLABEL:** Labels connected components in 2-D binary image.

A connected component in a binary image is a set of pixels that form a connected group, connected component labeling is the process of identifying the connected components in an image and assigning each one to a unique label, like in (Figure 5.7) below, where there are 3 separated shapes in the binary image.

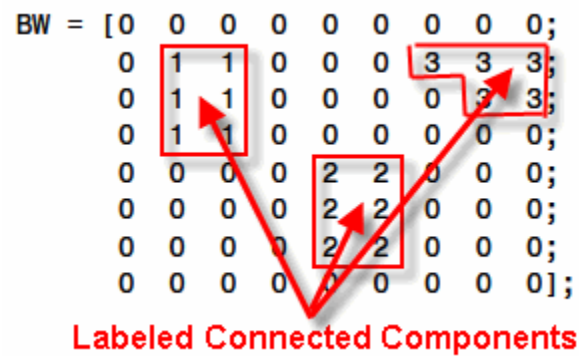


Figure 5.7: Label matrix

- **IMCROP:** crops a part of an image, as appears in (Figure 5.8).

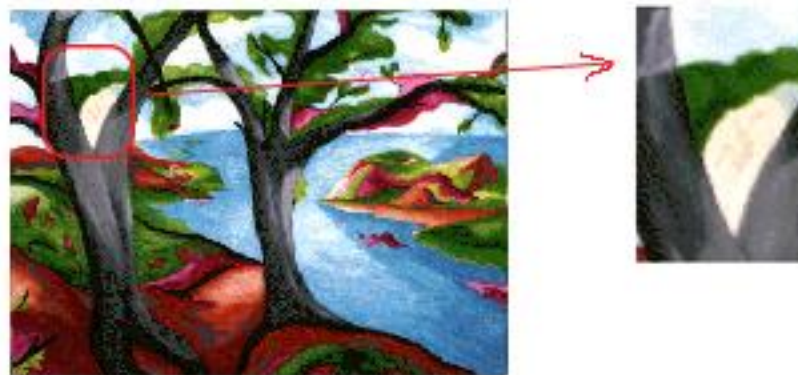


Figure 5.8: Cropping operation

5.3 Project Flowchart

The steps for the MATLAB simulation to identify the considered samples (words, numbers, signatures) from images can be summarized by the following flowchart:

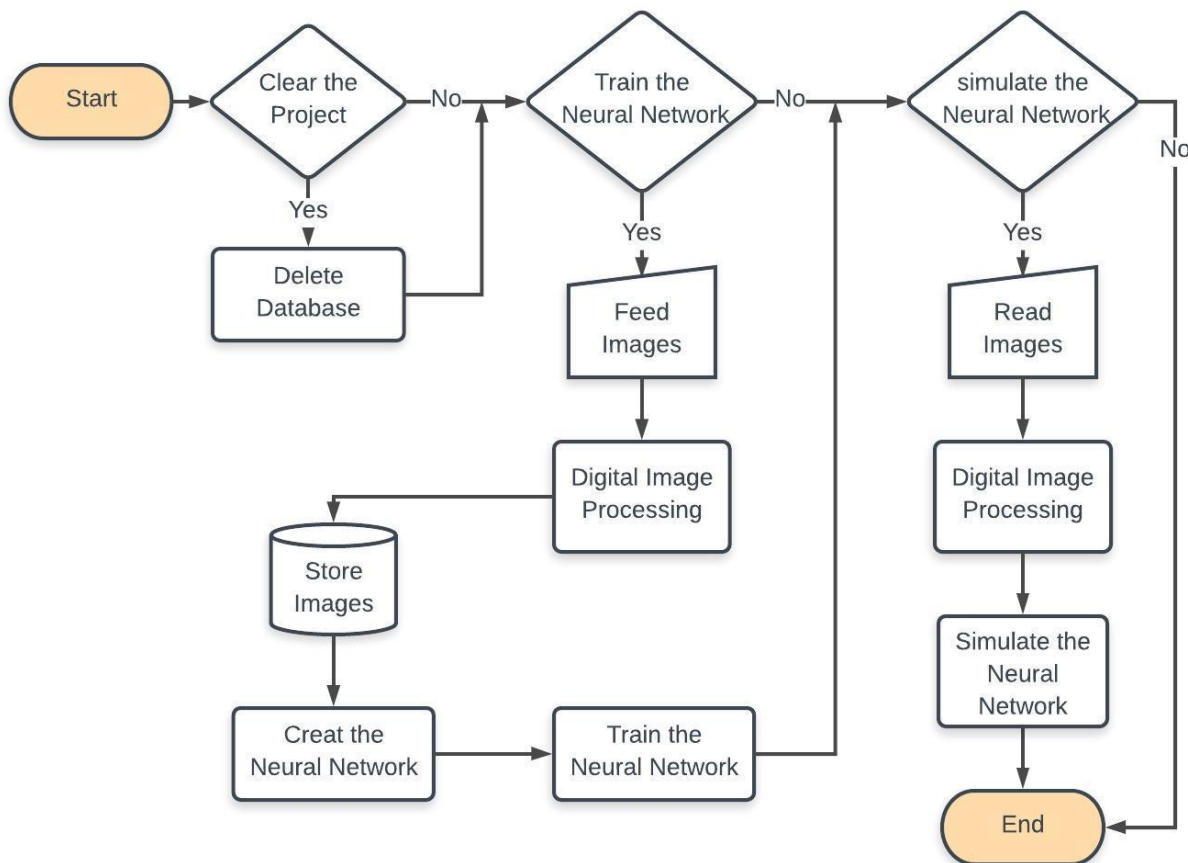


Figure 5.9: Project flowchart

5.4 Simulation & Results

1- Creating the graphical user interfaces using the GUIDE tool in MATLAB:

Link every button to a specific function so it achieves a desired purpose, such as Clear the project where the system checks the size of database file and delete it if it's not zero, Train the Neural Network to show an interface for setting dataset, creating and training the neural network, Run the project which linked to simulation function that runs the Network over testing objects.

All that have been done smoothly using GUIDE tool interface as shown in (figure 5.1) and Command Line interface. We can add identifying information or visual alert as seen in this project main interface (Figure 5.10).

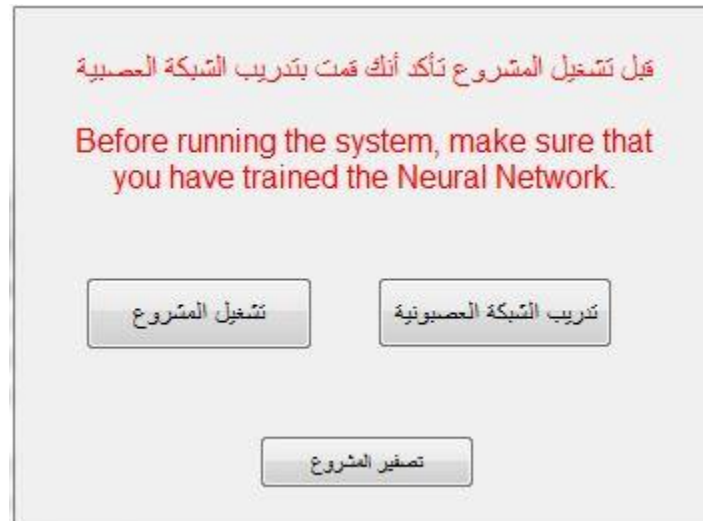


Figure 5.10: Project main interface

2- Implement Digital Image Processing (DIP) on the Images that hold the Data:

Feeding the system with multi-images to implement DIP on them then cut every sample alone and store it as a matrix in database. 4 samples have been used in this project to represent every information (Word, Number, Signature). We will use the 4 samples image in (Figure 5.11) to explain the effects of Digital Image Processing over, and to extract the error vectors which will be used as a part of Neural Network training input.

First MATLAB represents the image as a matrix with a size of how many pixels are in the image, every pixel represented in a number varies from 0 to 255, then converted to a binary image at 0.5 graythresh level, which means any element in the range of [0 – 127] consider as one (White), and any element in the range of [128 – 255] consider as zero (Black). (Figure 5.11) makes the operation more explicit.

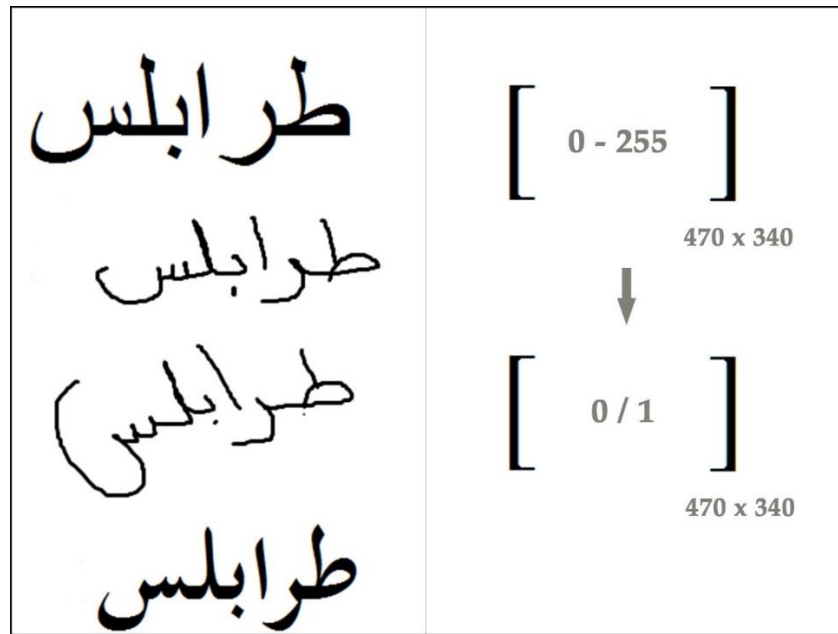


Figure 5.11: Samples Image representations

Then find the edges of the shapes in the image using EDGE function, so the result become as (Figure 5.12), after that we dilate the image both vertically and horizontality to clarify the borders of every shape in the image alone, the result shown in (Figure 5.13).

Last but not least remove any content still inside the shape to avoid any misconceptions, by filling the shapes, and that lead us to result in (Figure 5.14), lucky us in this example there is no much filler, so it depends on the shape of the image.



Figure 5.12: EDGE effect

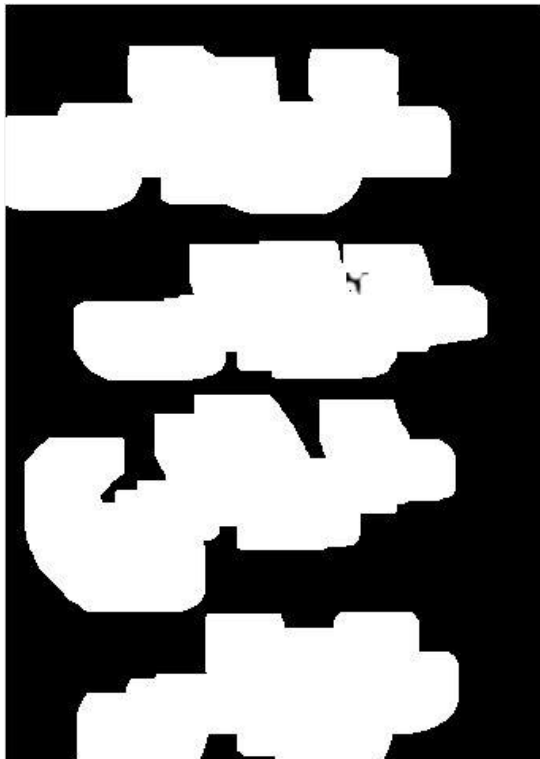


Figure 5.13: Dilated image

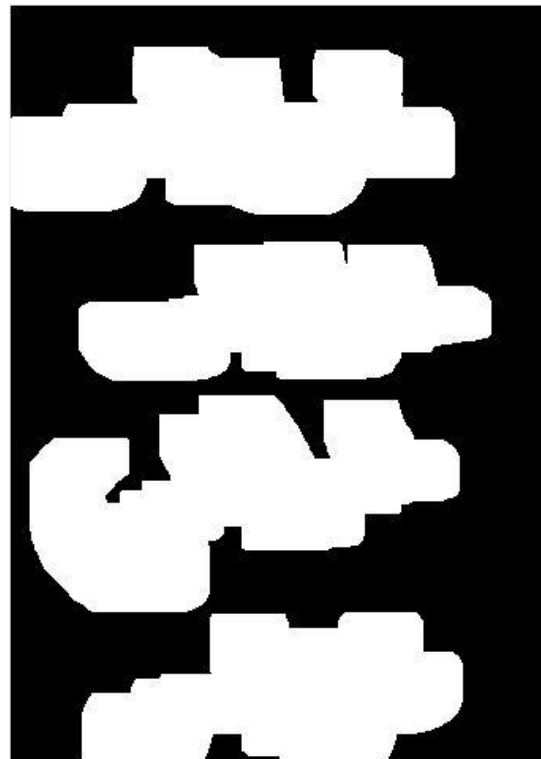


Figure 5.14: Filled dilated image

Now the image still a binary image, but after implementing the BWLABLE function, MATLAB leave zeros to stay as they are and collects every ones-collection connected together to represent a shape from the four shapes in the image and distinguishes them by change them to sorting numbers (1, 2, 3, 4), the result called Label matrix. (Figure 5.15) shows a label matrix made from a binary image that contains three shapes.

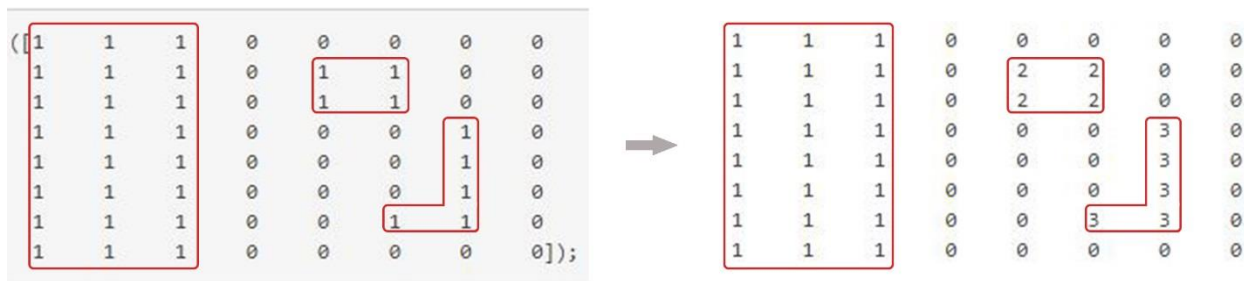


Figure 5.15: Labeling matrix for a binary image of 3 shapes

Since we have defined the borders of each shape, we draw a red rectangular around them, then move the red rectangles to the binary version of the original image as shown in (Figure 5.16) and crop every shape alone finally store it in a variable in database as a binary matrix with a link to text output.



Figure 5.16: Cropping samples from an image

Before moving to the neural network part, each shape in database should be cropped exactly at the edges as manifest in (Figure 5.17).

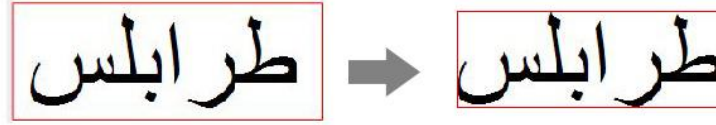


Figure 5.17: Cropping the edges

Then we resize it to a standard size, here I have chosen 100 x 290 pixels for this project, convert it to a vector of 290 elements horizontally by summing the elements of the same column together, then replace every element with the relative error of black pixels to white pixels in that column δx .

$$\delta x = \frac{\Delta x}{x} = \frac{|x_0 - x|}{x} = \frac{|x_0 - 100|}{100} \quad (15)$$

Where x_0 is the number of white pixels in that column, and x is the maximum value of white pixels quantity in one column which is equal to 100 (if every pixel is white in that column), Δx is the absolute error.

Transpose the vector vertically, then store the error vector of every sample from the four samples in one matrix, we repeat this process for each image of the dataset, we will end up with a matrix of 290 x 4*N, while N is the number of images stored in database, in this project I used 17 images as training subjects, so the matrix size was 290 x 68, this matrix will be the input of our neural network in the next stage. (Figure 5.18) and (Figure 5.19) will simplify the steps of this process in a visual way.



Figure 5.18: Extracting errors vector from a resized image

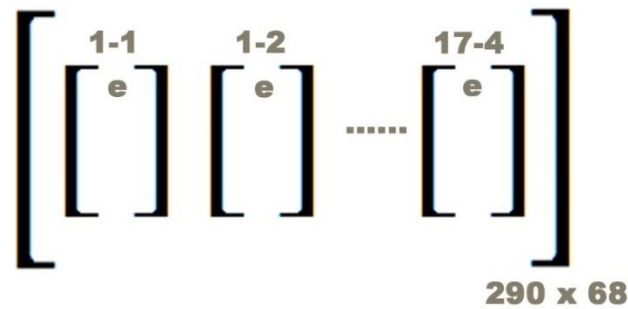


Figure 5.19: Collecting error vectors in one matrix

3- Creating and Training the Neural Network:

Now for the Network part, we use **newff** function to reach the Neural Network toolbox and Create a feed-forward network with three layers, 10 neurons for input layer, 10 neurons for hidden layer, and number of all expected outputs in the output layer, and since we stored only 17 images in the database, the last layer will contain 17 neurons. (Figure 5.20) displays created feed-forward neural network in MATLAB, (Figure 5.21) shows the project neural network layers in more details, where B is the bias of the neuron.

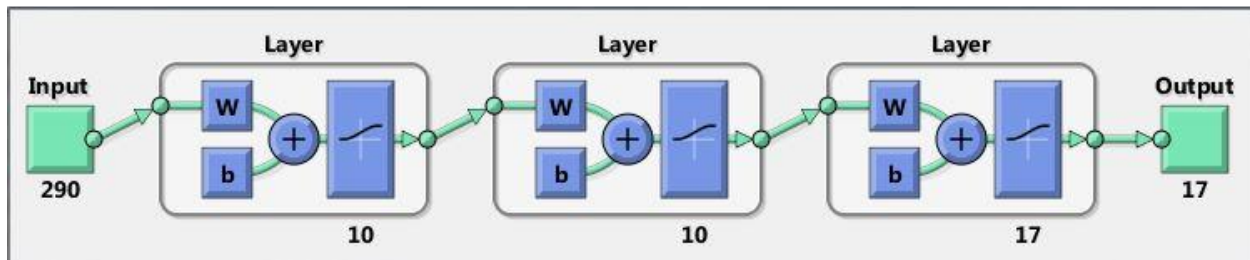


Figure 5.20: Created feed-forward neural network in MATLAB

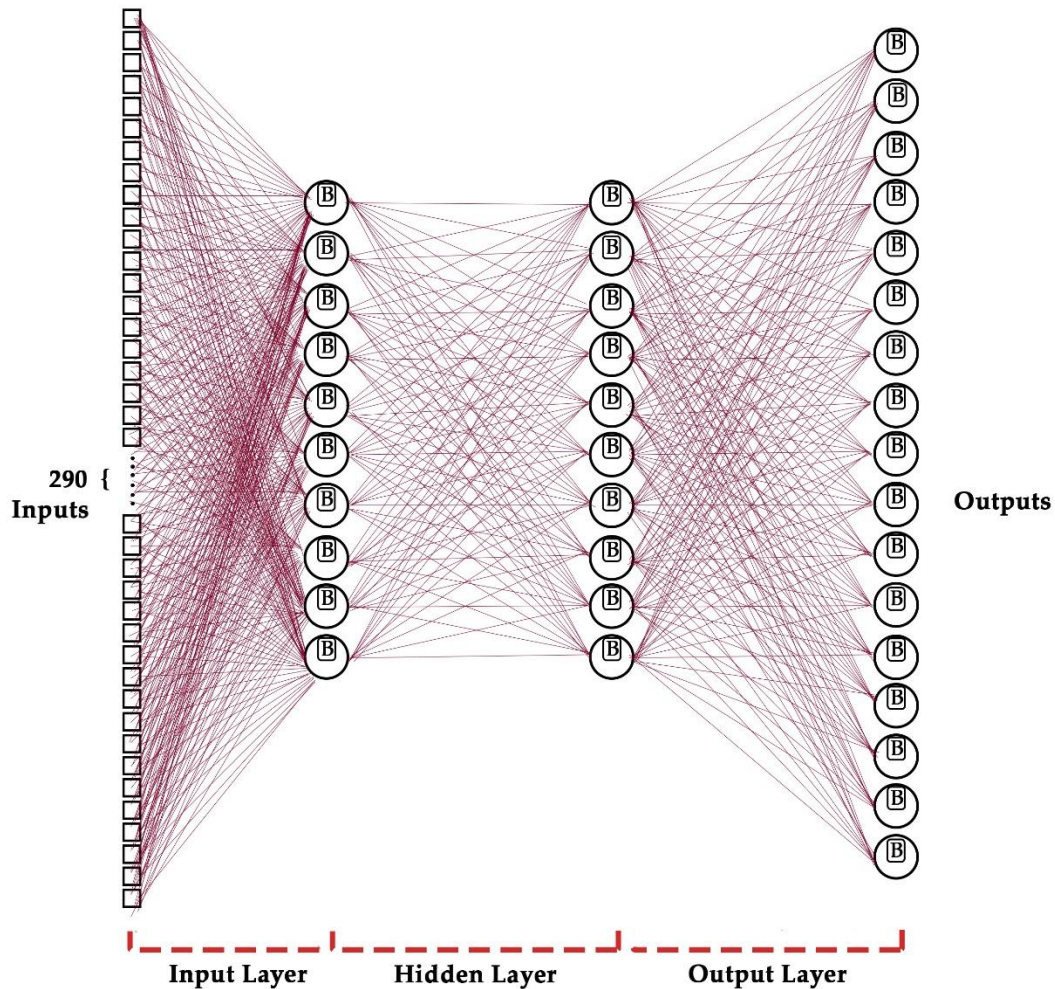


Figure 5.21: Project Neural Network structure

```
net = newff(minmax(input),[10 10 17],{'logsig','logsig','logsig'},'traingdx');
```

newff function requires four inputs and returns network object. The first input is a matrix of minimum and maximum values for each row of the input matrix that contains all the error vectors, so first input is a 290 x 2 matrix.

The second input is an array containing the sizes of each layer. The third input is a cell array containing the names of the transfer functions to be used in each layer, and here as we mentioned before we used sigmoid function as an activation function in all the layers. The final input contains the name of the training function to be used, which is here 'traingdx' Gradient descent with momentum and adaptive learning rate backpropagation training function.

We defined some properties of the net like using the Mean Square Error 'MSE' method to measure the network's performance according to the mean of squared errors, and maximum number of epochs, In the neural network terminology: one epoch = one forward pass and one backward pass of all the training examples, and other properties and specification such as a momentum constant (`net.trainParam.mc`) which added to slow the speed of the descent so that the search value does not fly back and forth across the minimum without stopping sufficiently near it.

```
net.performFcn = 'mse';
net.trainParam.show = 20;
net.trainParam.epochs = 2000;
net.trainParam.mc = 0.95;
```

Then we use **train** function to train the network on specifications defined before, the outputs that we get from this function are the trained Network as a network object and the training record which includes epochs numbers, Learning rate, performance functions and parameters, such as weights and biases.

Neural Network training tool interface will be shown in (Figure 5.22).

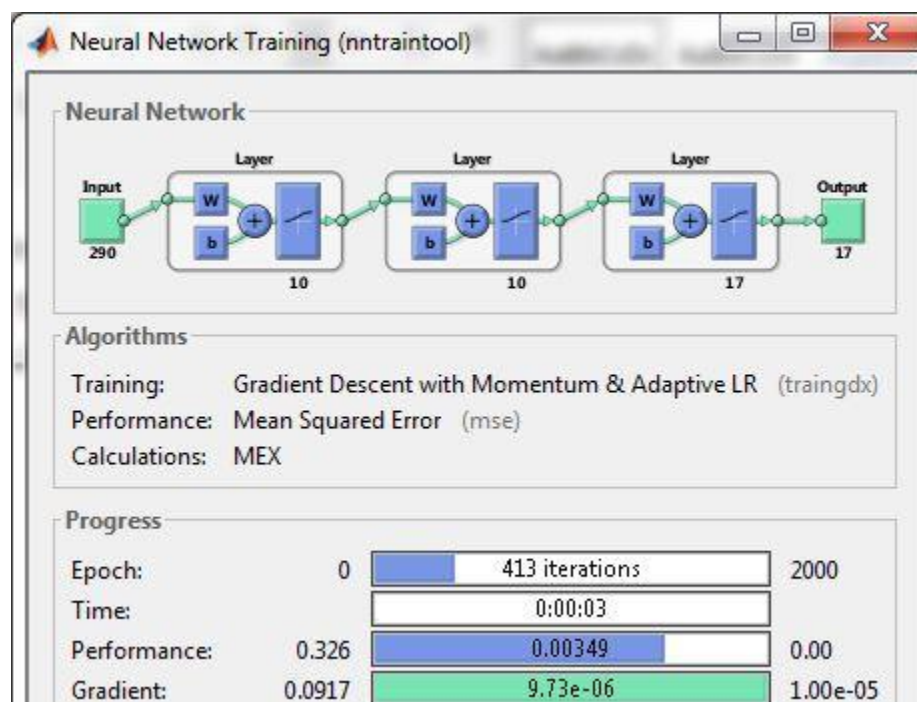


Figure 5.22: Neural Network training tool interface

Train function takes 3 inputs, the untrained network object that we've already created, the input and the target output.

The input here is the database error matrix shown in (Figure 5.19), and for the target we choose an ideal output, an identity matrix of $N \times N$, where N is the number of impossible outputs which is the number of the images in the database so the target output is a 17×17 Identity matrix.

After training the Neural Network we get the results of training such as Learning rate which shown in (Figure 5.23) the gradient in (Figure 5.24) the performance and epochs number in (Figure 5.25), the regression in (Figure 5.26) the weights of the hidden and output layers in (Figure 5.27-28) and the biases of every neuron in the neural network in (Figure 5.29).

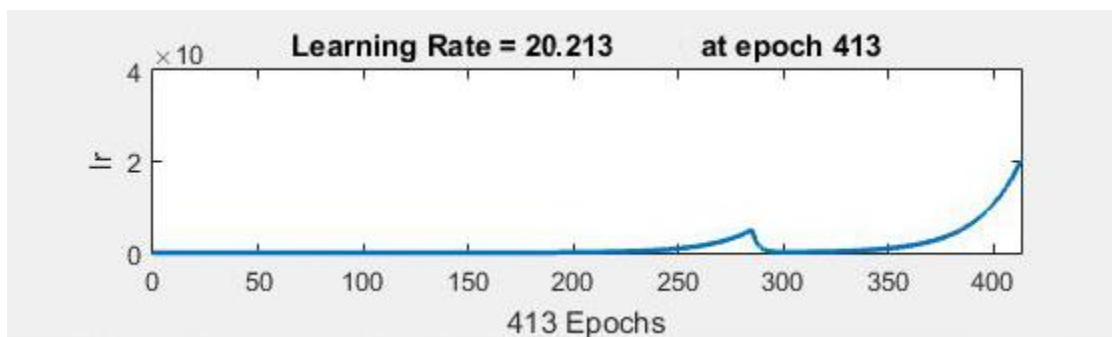


Figure 5.23: Learning rate over the epochs

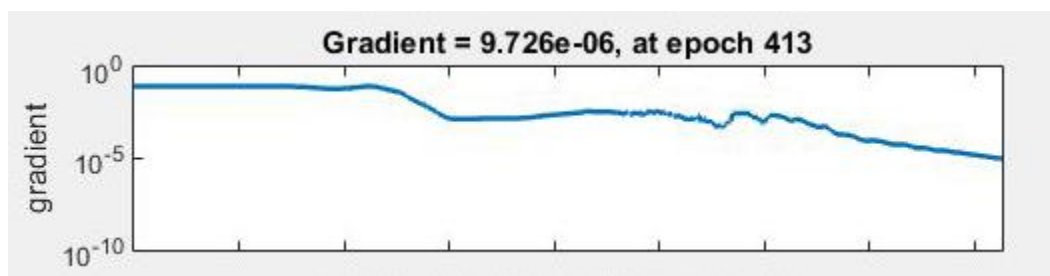


Figure 5.24: Gradient result

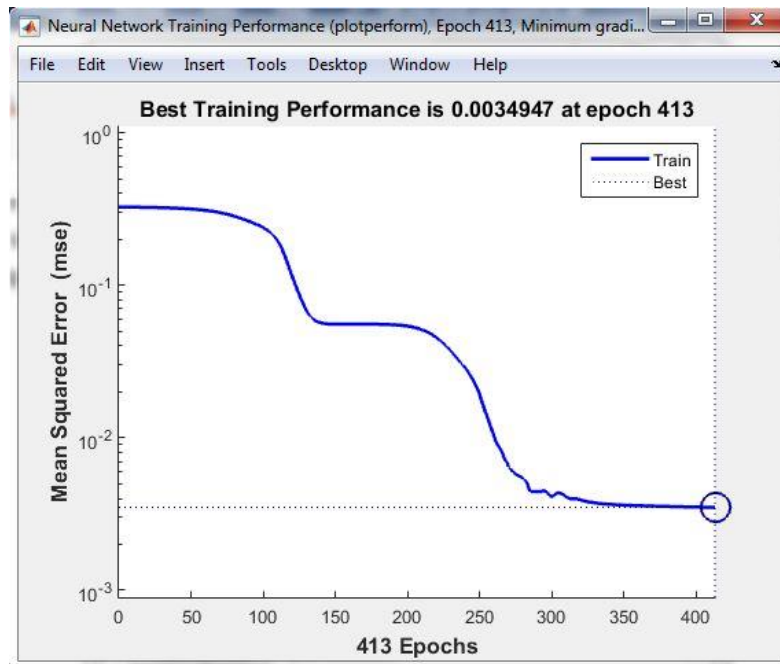


Figure 5.25: performance result

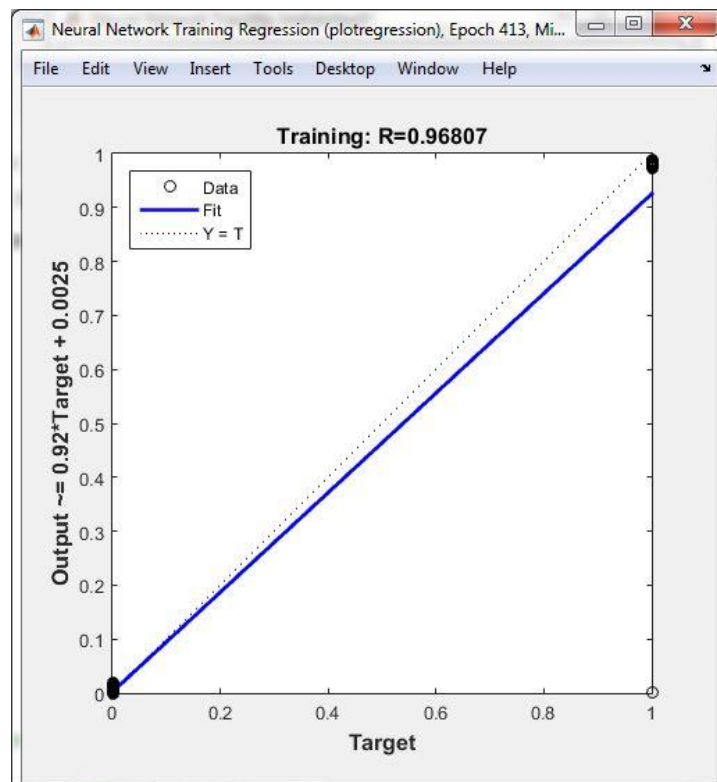


Figure 5.26: Regression result

	1	2	3	4	5	6	7	8	9	10
4	-4.1733	0.4902	2.1853	-2.4079	-4.8290	-3.3201	-0.2343	-0.8006	2.9967	1.8231
5	1.2112	0.8696	-3.3023	-2.8130	-1.4148	6.4695	0.9117	1.1055	-1.4200	0.2677
6	2.5102	0.6300	-1.6275	5.2640	-2.7834	-2.3718	-0.6392	1.4548	0.4396	0.6419
7	-0.2142	3.3230	-2.8762	3.0964	1.6439	0.2315	1.2287	-0.1289	-2.2379	-0.0225
8	2.8587	2.0847	-3.0847	-1.0608	-0.2379	-4.0163	0.7486	-4.8411	2.8994	1.5465
9	2.1066	-0.7660	3.6477	2.4653	2.8697	2.6166	1.6517	-1.9452	0.9041	-1.5196
10	-4.4648	0.7289	-1.3050	-4.2718	4.7934	-2.4830	3.2009	3.0615	1.4345	-0.2832

Figure 5.27: Adjusted weights of the hidden layer

	1	2	3	4	5	6	7	8	9	10
1	-4.0681	-0.1913	1.2119	3.4541	-2.5496	2.5541	-5.6909	-2.9749	-3.6513	4.2870
2	-4.7502	-1.0153	-4.3792	-2.5467	0.8911	2.7841	-0.5098	2.2847	-7.2554	-5.5001
3	4.6640	-4.5284	-1.7773	-1.3238	3.6705	0.1082	2.8090	-3.4536	-0.6886	-6.1265
4	4.0848	2.9030	-6.4654	-3.8713	-1.5028	-0.1830	3.6118	-2.0235	1.6730	1.5447
5	-4.8553	-3.7940	2.0956	3.4988	-3.1600	1.1225	-1.5374	4.2300	2.7996	-2.1060
6	-3.6756	-5.3288	-3.0374	4.2734	3.9491	-1.4198	-1.2287	-0.3206	-5.2728	2.7619
7	3.7010	-1.3144	0.2048	-4.2471	2.9233	-4.4547	-3.3135	-3.6727	-2.9882	5.0727
8	2.2256	0.4804	-3.7635	2.3827	-3.0767	1.8307	-5.3979	4.9610	-4.0012	-3.7523
9	-4.4649	-2.6371	0.0206	-3.6256	0.5369	-3.5622	0.0769	2.0402	-1.5333	-3.1449
10	-1.5829	1.1543	-0.4581	0.4701	-5.4005	3.4634	-4.8710	-6.1168	0.7782	-8.0165
11	3.1078	2.8822	-5.4603	0.9755	-1.0309	-5.4943	-4.2793	-5.1141	-2.5567	-2.4666
12	-0.3507	1.0430	-2.6501	2.0020	-7.6002	-4.4028	-2.2172	2.8833	2.4242	3.5044
13	-5.8083	-3.8489	-5.3746	-4.3946	-0.9185	0.5492	1.9111	3.1268	2.4528	1.6753
14	-5.0600	-3.3915	1.4666	-4.5027	0.1915	0.5421	-4.4748	-3.6781	1.5898	-4.3463
15	-5.8940	0.2550	-2.2583	-2.1037	-3.2768	-4.6001	3.0301	-5.3289	-0.6435	4.6693
16	1.9688	-8.0234	2.8347	0.0746	-4.8237	3.1924	-2.5674	-0.4913	-6.6145	-1.5916
17	-1.3912	4.1230	2.7174	-7.2499	-4.0209	0.5474	-1.5921	2.3220	1.6915	-2.8718

Figure 5.28: Adjusted weights of the output layer

1	-4.5112	0.3368	-4.0512
2	-4.2503	0.6269	2.9186
3	1.3227	-1.5052	-4.5332
4	-3.8807	0.9596	-7.1995
5	-2.6925	0.2956	-6.2777
6	-0.3814	0.3030	-1.4982
7	-3.5774	-0.2068	-2.7416
8	-1.0958	0.6937	-3.3608
9	7.2724	-2.3895	-1.0834
10	6.8690	-0.5370	1.3016
11			2.6971
12			-4.9186
13			-2.8620
14			3.2106
15			1.0803
16			-0.4055
17			-4.2503

Figure 5.29: Biases values for each neuron in the three layers

4- Finally, the recognition part:

From the project main interface, pressing **Run the project** button will take us to another interface, where we can choose the data image that we want to recognize, (Figure 5.30) will be used as a testing object in this part.



Figure 5.30: testing image

we apply the same Digital Image Processing techniques we used in step 2 with the same order to separate the shapes from one image, (Figure 5.31-32-33-34-35-36) show us the effects of DIP over the testing image.



Figure 5.31: Convert RGB testing image to grayscale

طرابلس
1 9 0 9 1

Figure 5.32: Convert the grayscale of testing image to black and white image



Figure 5.33: Finding the edges of black and white image



Figure 5.34: Dilating the black and white Image

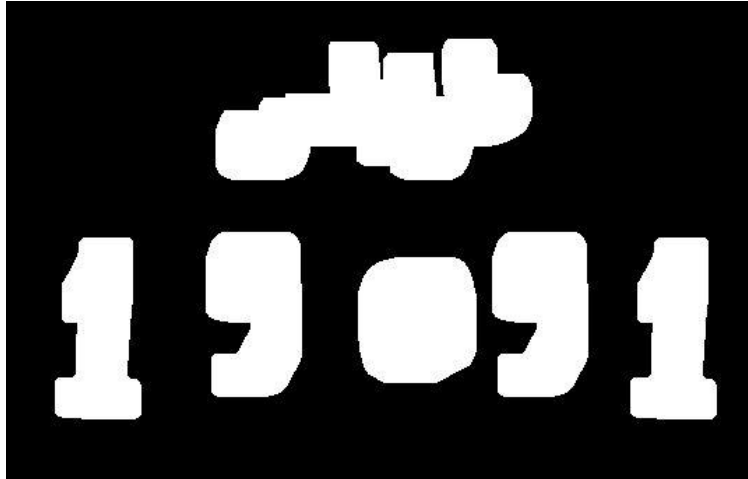


Figure 5.35: Filling dilated image regions and holes



Figure 5.36: Finding the limits of the shapes

get the error vector of every shape in the testing image, 290 elements in each vector of any (word, number, signature) found in the image.

Next, we simulate the trained network with each vector as an input then check the output, which will be a vector of 17 element represents the output layer, the index of the element closest to 1 refer to the desired object in the database, and every object in the database already linked to its text (string) value, after that the result transferred as a text to document.txt file, to show it to the user.

And to let the system be more precise, we make sure that there is only one value over 0.2 and all the other values are under 0.05 to find the desired index, otherwise if there was a match in more than one index (multiple element are over 0.2) or there is no matching at all (all output elements are under 0.2), we notify the user by printing a special symbol to tell him that the net failed to find this shape of data, here I chose a question mark '?' to represent failed data recognition. (Figure 5.37) shows the output of the network to one of the shapes, and we can see that the output at neuron 14 is the closest to one, while other outputs numbers are very small, this means that the testing shape here is very similar to the element with index (14) in the network database.

1	9.0651e-04
2	4.1007e-04
3	0.0151
4	1.9402e-06
5	0.0135
6	0.0021
7	0.0065
8	1.3132e-06
9	0.0033
10	0.0245
11	3.5035e-05
12	4.2758e-07
13	0.0010
14	0.9888
15	4.5715e-04
16	5.6994e-04
17	0.0062

Figure 5.37: The output layer at testing one of the shapes

5.5 Applications for this Project

Using the same concept and with the following of process steps discussed previously, we can gain the advantages of this project in multi applications such as:

Arabic handwritten words recognition

This project can be used to recognize Arabic words from Images taken by camera, and that will consider as a first step to translate that data to be understood by computer and then use it in different usages such as: translation from language to another, searching for a specific word, modifying text and save it in a database, etc.

(Figure 5.38) shows the result of using this project in this purpose, where the rectangular boxes show the detected words, and the notepad file is the program used to receive the text result.



Figure 5.38: Arabic handwritten words recognition

Signatures recognition

One of the uses of this project is to recognize signatures which can be used for security purposes, or to register the attendance at a facility, as appears in (Figure 5.39).

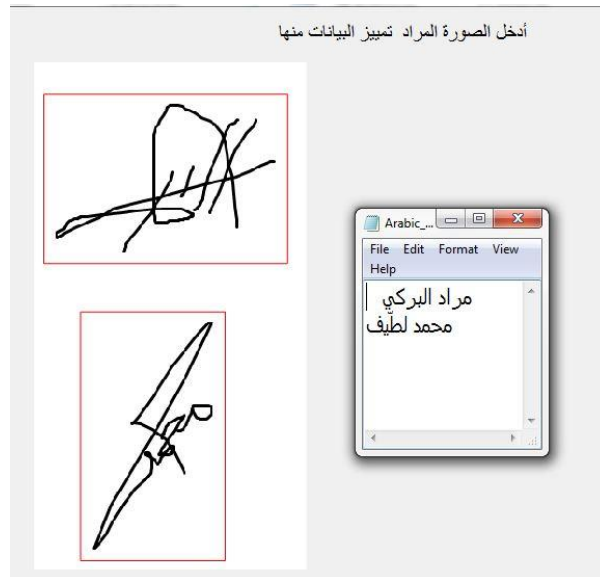


Figure 5.39: Signatures recognition

Vehicles registration plate recognition

One of the most Important application of this project is to recognize numbers, which is very helpful with vehicles plate numbers, mainly for security reasons, such as when the authorities try to track a vehicle or identify its owner from a footage has taken by camera. (Figure 5.40) shows the result of using this project in this purpose.

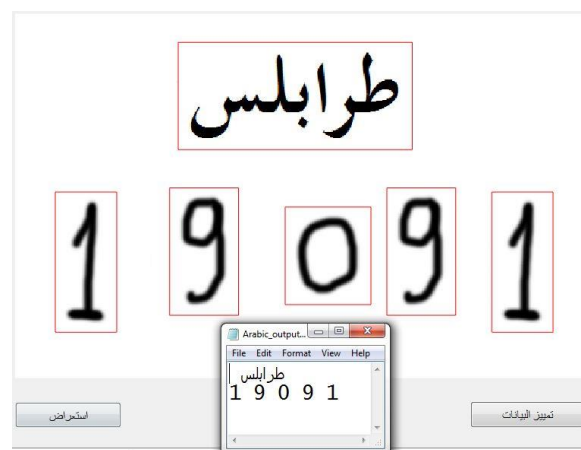


Figure 5.40: Vehicles registration plate recognition

Conclusion

We can summarize what we concluded from this project as follows:

- Neural Networks don't need to be programmed; they learn the program by themselves, based on the training by adjusting the weights and biases to get the most accurate output.
- Artificial Neural Networks have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex. The biggest issue, however, is that neural networks are “black boxes”, in which the user feeds in data and receives answers, but they don't have access to the exact decision-making process.
- Basically, having multiple layers (deep network) makes your network more accurate.
- Backpropagation requires supervision, that means input-output pairs must be provided during learning in order to compute the error between desired and actual output.
- Even though Neural Networks have a huge potential we will only get the best of them when they are integrated with computing, Artificial Intelligent and related subjects. Such as Digital Image Processing that used with Neural Network in this project to achieve the purpose of data recognition.
- This system is well suited for low-cost, real time hardware implementations; in this work, different samples are tested and the system showed very good results, but its performance would be much better if it has been trained on more than four samples for every data image.

Future Work

Finally, I would like to mention some ideas that can improve this project and make it more efficient and more practical depends on the application we are going to use it for, such as connecting the program directly to a traffic camera when we are attending to use it for vehicle registration recognition, or having a huge database with as much as possible amount of samples to make the system become more precise and efficient in handwritten words recognition then use it as a translation dictionary, we can also put the program on a server and design a mobile application to connect mobile camera to the server via the internet, so users can use it in foreigner countries to understand streets signs, or link this mobile application to an engineering calculator and use it to calculate equations by capturing images of them.

References

- [1] Dr. Pankaj Tiwari and Vikas Pathak. *Artificial Intelligence for All: An Abiding Destination*. India: Educreation publishing; 2018.
- [2] Ryszard S Michalski, Jaime G Carbonell, Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Berlin Heidelberg: Springer-Verlag; 1983.
- [3] Luke Dormehl, Michael Tyka. *what is an artificial neural network? Here's everything you need to know*. [online]. Available from: <https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/> [Accessed 5th Oct 2018].
- [4] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams. Learning Internal Representations by Error Propagation. Inc: James A. Anderson. Edward Rosenfeld (.eds): *Neurocomputing: foundations of research*. MIT Press Cambridge, MA, USA ©1988. P 673-695.
- [5] University of Tartu. UT: *Digital Image Processing*. [online]. Available from: <https://sisu.ut.ee/imageprocessing/book/1> [Accessed 5th Oct 2018].
- [6] University of Wisconsin-Madison: *A basic introduction to neural networks*, [online], Available from: <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html> [Accessed 5th Oct 2018].
- [7] Marvin Minsky, Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou. MIT Press, USA; 2017.
- [8] Igor Aleksander, Helen Morton. *An introduction to neural computing*. 2nd ed. London: International Thomson Computer Press; 1995.
- [9] Christos Stergiou and Dimitrios Siganos. *Neural Networks*. SURPRISE 96 Journal. [online]. Vol 4,1989, Imperial college London. Available from: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report [Accessed 5th Oct 2018].

- [10] Wikipedia: *Machine learning*. [online]. Available from: https://en.wikipedia.org/wiki/Machine_learning/ [Accessed 5th Oct 2018].
- [11] Ian F. Croall, John P. Mason (Eds.). *Industrial Applications of Neural Networks: Project ANNIE Handbook*. Volume 1 of Research Reports Esprit. Berlin Heidelberg: Springer Science & Business Media; 2012.
- [12] António Eduardo de Barros Ruano. *APPLICATIONS OF NEURAL NETWORKS TO CONTROL SYSTEMS*. United Kingdom Wales: University of Wales, Bangor; February, 1992.
- [13] Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*. 1989. 1(4), 541-551.
- [14] P.J. Werbos. *Backpropagation through time: what it does and how to do it*. Proceedings of the IEEE, Volume: 78, Issue: 10. Washington DC, USA; Oct 1990.
- [15] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. [online]. Available from: <http://ruder.io/optimizing-gradient-descent/> [Accessed 5th Oct 2018].
- [16] Sachin Joglekar. *Backpropagation for dummies*. [online]. Available from: <https://codesachin.wordpress.com/2015/12/06/backpropagation-for-dummies/> [Accessed 5th Oct 2018].
- [17] Assaad Moawad. *Neural networks and back-propagation explained in a simple way*. Medium [Online] 2018. Available from: <https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e> [Accessed 5th Oct 2018].
- [18] Winnie Y. Lo, Sarah M. Puchalski. *DIGITAL IMAGE PROCESSING*. Veterinary Radiology & Ultrasound, Vol. 49, No. 1, Supp. 1. 2008. pp S42–S47.
- [19] Kenneth R. Castleman. *Digital Image Processing*. Prentice Hall Press Upper Saddle River, NJ, USA ©1996.

- [20] B. Chitradevi, P. Srimathi. *An Overview on Image Processing Techniques*. International Journal of Innovative Research in Computer and Communication Engineering. [online]. Available from: <http://www.rroij.com/open-access/an-overview-on-image-processing-techniques.php?aid=47175> [Accessed 5th Oct 2018].
- [21] Ernest Hall. *Computer Image Processing and Recognition*. Academic Press; 1979.
- [22] M. Mansourpour, M.A. Rajabi, J.A.R. Blais. *EFFECTS AND PERFORMANCE OF SPECKLE NOISE REDUCTION FILTERS ON ACTIVE RADAR AND SAR IMAGES*. ISPRSL, XXXVII/1-W41.
- [23] Young GiByun, You Kyung Han, Tae ByeongChae. *A Multispectral Image Segmentation Approach for Object-based Image Classification of High-Resolution Satellite Imagery*. KSCE; 2012.
- [24] Gaurav Kumar, Pradeep Kumar Bhatia. *A Detailed Review of Feature Extraction in Image Processing Systems*. Fourth International Conference on Advanced Computing & Communication Technologies Pages 5-12, IEEE Computer Society Washington, DC, USA ©2014.
- [25] Rama Chellappa. *Digital Image Processing*. 2nd Edition, IEEE Computer Society Press; 1992.
- [26] MathWorks corporation. [Online]. Available from: <https://www.mathworks.com/> [Accessed 5th Oct 2018].

Appendix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           *Digital Image Processing*
%           DIP at storing the samples
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global selected_col;
global selected_ln;
global all_img_1;
global all_img_2;
global all_img_3;
global all_img_4;
global all_words;
new_word = get(handles.edit1, 'string');
if size(new_word,2) > 0
    try
        trimming = handles.S;
        Igray = rgb2gray(trimming);
        Ibw = im2bw(Igray,graythresh(Igray));
        Iedge = edge(uint8(Ibw));
        %-----
        se = strel('Line',25,0);
        sel = strel('Line',20,90);
        Iedge2 = imdilate(Iedge, se);
        Iedge3 = imdilate(Iedge2, sel);
        Ifill = imfill(Iedge3, 'holes');
        [Ilabel num] = bwlabel(Ifill);
        Iprops = regionprops(Ilabel);
        Ibox = [Iprops.BoundingBox];
        [y,x]=size(Ibox);
        x=x/4;
        Ibox = reshape(Ibox,[4 x])
        for cnt = 1:size(Ibox,2)
            rectangle('position',Ibox(:,cnt),'edgecolor','r');
            img{cnt} = imcrop(trimg,Ibox(:,cnt));
        end
        if selected_ln ~= size(img,2)/selected_col
            Error_more_than_4_words;
        else
            all_img_1{size(all_img_1,2)+1} = img{1};
            all_img_2{size(all_img_2,2)+1} = img{2};
            all_img_3{size(all_img_3,2)+1} = img{3};
            all_img_4{size(all_img_4,2)+1} = img{4};
            all_words{size(all_words,2)+1} = {new_word};
            %Word Added Successfully
            set(handles.edit1, 'string','');
            wrod_successfully_added;
        end
    catch ME
        Error_no_word;
    end
end
```

```

else
    Error_no_word;
end
%-----

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% DIP techniques before creating the network
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global selected_ln;
global all_img_1;
global all_img_2;
global all_img_3;
global all_img_4;
global all_words;
counter = 1;
out = [];
for cnt = 1:size(all_img_1,2);
    bw2 = charcrop(all_img_1{cnt});
    charvec = figresize(bw2);
    out(:,counter) = charvec;
    counter = counter + 1;
end
for cnt = 1:size(all_img_2,2);
    bw2 = charcrop(all_img_2{cnt});
    charvec = figresize(bw2);
    out(:,counter) = charvec;
    counter = counter + 1;
end
for cnt = 1:size(all_img_3,2);
    bw2 = charcrop(all_img_3{cnt});
    charvec = figresize(bw2);
    out(:,counter) = charvec;
    counter = counter + 1;
end
for cnt = 1:size(all_img_4,2);
    bw2 = charcrop(all_img_4{cnt});
    charvec = figresize(bw2);
    out(:,counter) = charvec;
    counter = counter + 1;
end

if size(out,2) > 0
    P = out(:,1:size(all_img_1,2)*selected_ln);
    T = [];
    for xxx = 1:selected_ln
        T = [T eye(size(all_img_1,2))];
    end

%-----

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DIP functions to find boundaries and crop the edges
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function bw2 = charcrop(bw)

% Find the boundary of the image
[y2temp x2temp] = size(bw);
x1=1;
y1=1;
x2=x2temp;
y2=y2temp;

%left side
cntB=1;
while (sum(bw(:,cntB))==y2temp)
    x1=x1+1;
    cntB=cntB+1;
end

%right side
cntB=1;
while (sum(bw(cntB,:))==x2temp)
    y1=y1+1;
    cntB=cntB+1;
end

%up side
cntB=x2temp;
while (sum(bw(:,cntB))==y2temp)
    x2=x2-1;
    cntB=cntB-1;
end

%down side
cntB=y2temp;
while (sum(bw(cntB,:))==x2temp)
    y2=y2-1;
    cntB=cntB-1;
end

% Crop the char
bw2=imcrop(bw,[x1,y1,(x2-x1),(y2-y1)]);

%-----

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%resize the image and find the error vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sevec = figresize(bw2)
% This function take the cropped binary image and resize it to 100 x 290
% char representation as single vector.

bw_7050 = imresize(bw2,[100,290]);
for cnt=1:10
    for cnt2=1:29
        Atemp = sum(bw_7050((cnt*10-9:cnt*10),(cnt2*10-9:cnt2*10)));
        sevec((cnt-1)*29+cnt2) = sum(Atemp);
    end
end
sevec = ((100-sevec)/100);
sevec = sevec';
%-----

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                *Training Part*
%% Creating and training of the Neural Network
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

alphabet = P;
targets = T;

[R,Q] = size(alphabet);
[S2,Q] = size(targets);
S1 = 10;
net = newff(minmax(alphabet),[S1 10 S2],{'logsig' 'logsig' 'logsig'
},'traingdx');
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;
net.performFcn = 'mse';
net.trainParam.show = 20;
net.trainParam.epochs = 2000;
net.trainParam.mc = 0.95;
P = alphabet;
T = targets;
[net,tr] = train(net,P,T);

%% Testing the Neural Network
    assignin('base','net',net);
    guidata(hObject, handles);
    %save NN ALL

save('DataBase.mat','net','all_img_1','all_img_2','all_img_3','all_img_4','al
l_words','-mat');
    clear;
    clear global;
    close;
end
%-----

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           *Simulation Part*
%
%   DIP & Net simulation for Recognition
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global net;
global all_words;
try
    I = handles.S;
    %Threshold
    Igray = rgb2gray(I);
    Ibw = im2bw(Igray,graythresh(Igray));
    Iedge = edge(uint8(Ibw));
    %-----
    se = strel('Line',25,0);
    se1 = strel('Line',20,90);
    Iedge2 = imdilate(Iedge, se);
    Iedge3 = imdilate(Iedge2, se1);
    Ifill = imfill(Iedge3,'holes');
    [Ilabel num] = bwlabel(Ifill);
    Iprops = regionprops(Ilabel);
    Ibox = [Iprops.BoundingBox];
    [y,x]=size(Ibox);
    x=x/4;
    Ibox = reshape(Ibox,[4 x]);

    Fbox = [];
    line = [];
    counter = 0;
    for qq =1:size(Ibox,2)
        Tbox = [];
        min = 1024;
        avg_height = 0;
        for j = 1:size(Ibox,2)
            if min > Ibox(2,j)
                min = Ibox(2,j);
                counter = counter + 1;
            end
            avg_height = avg_height + Ibox(4,j);
        end
        avg_height = avg_height / size(Ibox,2);

        for j = 1:size(Ibox,2)
            if abs(min - Ibox(2,j)) < avg_height
                Tbox = [Tbox Ibox(:,j)];
                line = [line counter];
                Ibox(2,j) = 2024;
            end
        end
        if size(Tbox,1) > 0
            Tbox = sortrows(Tbox',-1)';
            Fbox = [Fbox Tbox];
        end
    end
end

```

```

end
Ibox = Fbox;
fid = fopen('Arabic_output.txt', 'w', 'n', 'UTF-8');
current_line = line(1);
for cnt = 1:size(Ibox,2)
    rectangle('position',Ibox(:,cnt),'edgecolor','r');
    img{cnt} = imcrop(Ibw,Ibox(:,cnt));
    %pre-process
    bw2 = charcrop(img{cnt});
    %-----
    %Feature Extraction
    charvec = figresize(bw2);
    %-----
    %Recognize
    result = sim(net,charvec);
    if sum(ones(size(result))-result<=0.05) == 1 && sum(result >= 0.2) == 1
        [val, num] = max(result);
        str = char(all_words{num});

else
    str = '?';
end
if current_line == line(cnt)
    fwrite(fid, ' ', 'char');
else
    str1 = char(13);
    fwrite(fid, str1, 'char');
    str2 = char(10);
    fwrite(fid, str2, 'char');
    current_line = line(cnt);
end
fwrite(fid, str, 'char');

%-----
end
fclose(fid);
dos('notepad Arabic_output.txt &');
guidata(hObject, handles);
catch ME
    Error_no_word;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```